

Luis Miguel Fernandez Diaz_ Jose Carlos Tapia Riv...

DESARROLLO DE UN APLICATIVO DE RECOLECCIÓN DE INFORMACIÓN INTEGRADO CON BIM PARA LA GESTIÓN DE...

 Quick Submit Quick Submit Universidad Nacional de Jaen

Detalles del documento

Identificador de la entrega

trn:oid:::1:3515204410

Fecha de entrega

23 mar 2026, 2:50 p.m. GMT-5

Fecha de descarga

23 mar 2026, 2:53 p.m. GMT-5

Nombre del archivo

INFORME_FINAL_AL_100_-MODIFICADO.pdf

Tamaño del archivo

21.2 MB

184 páginas

33.403 palabras

208.832 caracteres

2% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Coincidencias menores (menos de 15 palabras)

Fuentes principales

- 2%  Fuentes de Internet
- 0%  Publicaciones
- 1%  Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.º de alertas de integridad para revisión

No se han detectado manipulaciones de texto sospechosas.

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.

Fuentes principales

2%	Fuentes de Internet
0%	Publicaciones
1%	Trabajos entregados (trabajos del estudiante)

Fuentes principales

Las fuentes con el mayor número de coincidencias dentro de la entrega. Las fuentes superpuestas no se mostrarán.

1	Internet	repositorio.unj.edu.pe	<1%
2	Internet	alicia.concytec.gob.pe	<1%
3	Internet	repository.udistrital.edu.co	<1%
4	Internet	hdl.handle.net	<1%
5	Internet	repositorio.ucv.edu.pe	<1%
6	Trabajos del estudiante	Universidad Privada del Norte	<1%
7	Trabajos del estudiante	Universidad Andina del Cusco	<1%
8	Trabajos del estudiante	Universidad Privada Antenor Orrego	<1%
9	Trabajos del estudiante	Corporación Universitaria Remington	<1%
10	Trabajos del estudiante	Universidad Nacional de Jaen	<1%
11	Internet	catalogobiblioteca.ufps.edu.co	<1%

12	Internet	upc.aws.openrepository.com	<1%
13	Internet	repositorio.uide.edu.ec	<1%
14	Internet	repositorio.upn.edu.pe	<1%
15	Internet	www.coursehero.com	<1%
16	Trabajos del estudiante	Universidad Católica San Pablo	<1%
17	Trabajos del estudiante	Universidad San Ignacio de Loyola	<1%

UNIVERSIDAD NACIONAL DE JAÉN
FACULTAD DE INGENIERÍA
CARRERA PROFESIONAL DE INGENIERÍA CIVIL



**DESARROLLO DE UN APLICATIVO DE RECOLECCIÓN
DE INFORMACIÓN INTEGRADO CON BIM PARA LA
GESTIÓN DEL MANTENIMIENTO DE LA INSTITUCIÓN
EDUCATIVA RAMÓN CASTILLA Y MARQUESADO,
JAÉN – 2025**

**INFORME FINAL DE TESIS PARA OPTAR EL TÍTULO
PROFESIONAL DE INGENIERO CIVIL**

Autores: **Bach. Luis Miguel Fernandez Diaz**
 Bach. Jose Carlos Tapia Rivera

Asesor: **Mg. Billy Alexis Cayatopa Calderón**

Línea de Investigación: LI_IC_ 03 Gestión y Tecnología

JAÉN – PERÚ, MARZO, 2026

ÍNDICE GENERAL

	Pág.
RESUMEN	x
ABSTRACT	xi
I. INTRODUCCIÓN	12
1.1. Descripción de la realidad Problemática	12
1.2. Justificación	13
1.2.1. Justificación Científica	13
1.2.2. Justificación Económica	13
1.2.3. Justificación Social	14
1.2.4. Justificación ambiental	14
1.3. Hipótesis	15
1.4. Objetivos	15
1.4.1. Objetivo general.....	15
1.4.2. Objetivos específicos	15
1.5. Antecedentes de la investigación.....	16
1.5.1. Internacionales.....	16
1.5.2. Nacionales.....	17
1.5.3. Regional y/o local	19
II. MATERIAL Y MÉTODOS	20
2.1. Ubicación geográfica	20
2.2. Población, muestra y muestreo	20
2.2.1. Población	20
2.2.2. Muestra	22
2.2.3. Muestreo	24
2.3. Metodología.....	24
2.3.1. Método.....	24
2.3.2. Técnicas	24
2.3.3. Instrumentos	25
2.3.4. Procedimientos de recolección de datos	25
2.3.4.1. FASE 1: Diagnóstico de la situación actual	25
2.3.4.2. FASE 2: Diseño del aplicativo de recolección de información.....	26
2.3.4.3. FASE 3: Modelado BIM de la Institución Educativa.....	97

2.3.4.4. FASE 4: Integración del Revit con el Modelo BIM.....	118
2.3.4.5. FASE 5: Implementación y Evaluación del Aplicativo	132
2.3.4.6. FASE 6: Integración del Aplicativo con el Modelo BIM	137
2.3.5. Análisis de datos	145
III. RESULTADOS	146
3.1. Diseñar y desarrollar un aplicativo para la recolección de información para la gestión del mantenimiento en la Institución Educativa Ramón Castilla y Marquesado. ..	146
3.2. Implementar y evaluar el aplicativo en la recolección de datos de infraestructura para la gestión y mantenimiento en la Institución Educativa Ramón Castilla y Marquesado.	155
3.3. Integrar la información recopilada del aplicativo mediante el modelo BIM para optimizar la gestión del mantenimiento en la Institución Educativa Ramón Castilla y Marquesado.	158
IV. DISCUSIÓN	169
V. CONCLUSIONES Y RECOMENDACIONES	177
5.1. Conclusiones.....	177
5.2. Recomendaciones	178
REFERENCIAS BIBLIOGRÁFICAS	180
AGRADECIMIENTO	184
DEDICATORIA	185
ANEXOS	186

ÍNDICE DE TABLAS

Tabla 1. Conversión de Condición Física	116
Tabla 2. Conversión de Riesgo Normativo.....	116
Tabla 3. Conversión de Estado Operativo	116
Tabla 4. Clasificación Final del Nivel de Criticidad	116

ÍNDICE DE FIGURAS

Figura 1. IE. N°16001 Ramón Castilla y Marquesado	20
Figura 2. Primer piso del bloque I de la IE. N°16001 Ramón Castilla y Marquesado ..	21
Figura 3. Segundo piso del bloque I de la IE. N°16001 Ramón Castilla y Marquesado	22
Figura 4. Versión de Flutter instalada en el sistema	25
Figura 5. Versión de Android Studio utilizada en el desarrollo é.....	26
Figura 6. Versión de Visual Studio Code utilizada en el desarrollo.....	26
Figura 7. Creación del proyecto en la consola de Firebase para la configuración de los servicios en la nube.....	28
Figura 8. Configuración del servicio Firebase Authentication para el control de acceso de usuarios del aplicativo.....	28
Figura 9. Configuración de la base de datos Cloud Firestore para el almacenamiento de información del aplicativo.	29
Figura 10. Configuración de FlutterFire para la integración del aplicativo Flutter con los servicios de Firebase.	30
Figura 11. Creación y configuración del proyecto en la plataforma Supabase para el almacenamiento de archivos del aplicativo	31
Figura 12. Configuración del bucket de almacenamiento en Supabase para la gestión de imágenes y archivos del aplicativo	31
Figura 13. Obtención de URL y Configuración en Flutter	32
Figura 14. Estructura interna del módulo de autenticación (auth/)	34
Figura 15. Módulo principal – dashboard (Estructura interna del módulo)	36
Figura 16. Estructura interna del módulo Productos (Gestión de Activos)	38
Figura 17. Lista_productos_screen.dart – Lectura (Read).....	39
Figura 18. Agregar_producto_screen.dart – Creación (Create).....	40
Figura 19. Editar_producto_screen.dart – Actualización (Update)	41
Figura 20. Reportes_del_producto_screen.dart – Relación Uno a Muchos	42
Figura 21. Integración con Supabase (Evidencias).....	43
Figura 22. Estructura interna del módulo Reportes.	44
Figura 23. Reportes_screen.dart.	45
Figura 24. Categorias_reporte_screen.dart	46
Figura 25. Lista_reportes_por_categoria_screen.dart.....	47

Figura 26. Generación_reporte_screen.dart.....	48
Figura 27. Seleccion_producto_reporte_screen.dart	49
Figura 28. Detalle_reporte_screen.dart.....	50
Figura 29. Reporte_card.dart	51
Figura 30. Estructura interna del módulo Parámetros	52
Figura 31. Parametros_import_screen.dart.....	54
Figura 32. Parametros_screen.dart.....	55
Figura 33. Parámetros_templates.dart	56
Figura 34. Párametros_viewer_screen.dart.....	57
Figura 35. Implementación del módulo de escaneo QR.	59
Figura 36. Análisis Técnico del Código	60
Figura 37. Estructura interna del módulo Services (Lógica de Negocio).....	61
Figura 38. Activo_id_helper.dart.....	62
Figura 39. Audit_service.dart	63
Figura 40. Auth_user_profile_service.dart	64
Figura 41. Categorías_service.dart3	65
Figura 42. Parámetros_import_service.dart.....	66
Figura 43. Parámetros_schema_service.dart	67
Figura 44. Excel_export_service.dart1	68
Figura 45. File_save_service.dart	69
Figura 46. Pdf_service.dart.....	70
Figura 47. Schema_service.dart.....	71
Figura 48. Módulo auditoría	72
Figura 49. Módulo usuarios	73
Figura 50. Estructura interna del módulo Shared.	74
Figura 51. Disciplinas_categorías.dart.....	75
Figura 52. Disciplinas_config.dart1.....	76
Figura 53. Text_formatters.dart1	77
Figura 54. Inicialización de Firebase en el archivo main.dart	78
Figura 55. Proceso de Inicio de Sesión (Login).....	79
Figura 56. Creación de Activo (Create).....	81
Figura 57. Lectura de Activo (Read)	82
Figura 58. Actualización de Activo (Update).....	83

Figura 59. Implementación del Escáner QR.....	84
Figura 60. Carga de Imágenes en Almacenamiento Externo.....	85
Figura 61. Ejecución del aplicativo en entorno Web.....	86
Figura 62. Ejecución del flujo de compilación automatizado en GitHub Actions.	87
Figura 63. Este archivo constituye el paquete instalable del aplicativo para dispositivos iOS.	89
Figura 64. Instalación del aplicativo en dispositivo iOS utilizando Sideloadly.n	90
Figura 65. Estado del entorno de desarrollo mediante flutter doctor.....	91
Figura 66. Toma de medidas de la infraestructura.....	92
Figura 67. Segundo piso	93
Figura 68. Creación de carpetas.....	94
Figura 69. Inicio del modelo de estructura	95
Figura 70. Configuración previa	95
Figura 71. Modelo en 2D.....	96
Figura 72. Modelado de columnas.....	96
Figura 73. Modelado de vigas.....	97
Figura 74. Modelado de losa, muros y parapetos	97
Figura 75. Modelado de escalera	98
Figura 76. Modelo de estructuras vinculado al archivo Revit de arquitectura	98
Figura 77. Codificación del tarrajeo muros	99
Figura 78. Codificación de suelos (tarrajeo en losas).....	100
Figura 79. Codificación de pintura	100
Figura 80. Codificación de suelos (contrapisos).....	101
Figura 81. Codificación de cobertura	102
Figura 82. Codificación de puertas	102
Figura 83. Codificación de ventana frontal alta.....	103
Figura 84. Codificación de ventana posteriores.....	104
Figura 85. Codificación de ventana de escalera	104
Figura 86. Modelo de estructura vinculado al archivo Revit de estructuras y arquitectura	105
Figura 87. Inserción de tomacorriente	106
Figura 88. Inserción de luminarias led tipo bombillas.....	106
Figura 89. Inserción de luminarias tipo panel.....	107

Figura 90. Inserción de luminaria simple circular empotrada	108
Figura 91. Inserción de tablero eléctrico	109
Figura 92. Inserción de silla unipersonal	109
Figura 93. Inserción de escritorio docente	110
Figura 94. Inserción de silla docente	111
Figura 95. Inserción de pizarra acrílica	111
Figura 96. Inserción de armario pequeño	112
Figura 97. Inserción de armario grande	112
Figura 98. Inserción de TV	113
Figura 99. Creación del archivo de parámetros compartidos (.txt) en Autodesk Revit para interoperabilidad BIM.....	117
Figura 100. Organización de parámetros compartidos en grupos “Datos de Identidad” y “Gestión del Mantenimiento”	118
Figura 101. Integración de parámetros compartidos como parámetros de proyecto y su vinculación a categorías de arquitectura, eléctricas y mobiliario	119
Figura 102. Extracción de elementos del modelo BIM mediante nodos “Categories” y “All Elements of Category” en Dynamo.	120
Figura 103. Configuración del script en Dynamo para la lectura y estructuración de parámetros del modelo BIM.	121
Figura 104. Configuración del script en Dynamo para la consolidación tabular y exportación de parámetros a base de datos externa.	122
Figura 105. Esquema integral del proceso de interoperabilidad BIM–Excel mediante Dynamo.....	123
Figura 106. Replicación del script en Dynamo para múltiples categorías y disciplinas del modelo BIM.	124
Figura 107. Archivos Excel generados tras la ejecución del script de extracción de parámetros en Dynamo.	125
Figura 108. Importación de la base de datos Excel al aplicativo de gestión de mantenimiento.	126
Figura 109. Proceso de análisis e importación de base de datos Excel en el aplicativo de mantenimiento.	127
Figura 110. Inicio del levantamiento en campo con el aplicativo	128
Figura 111. Identificación de los activos mediante código QR.....	129

Figura 112. Generación de reporte de cada activo mediante el aplicativo	130
Figura 113. Generación de reportes	131
Figura 114. Guardado de reportes generados	132
Figura 115. Importación final de los parámetros gestionados con el aplicativo.....	133
Figura 116. Creación del proyecto en speckle, herramienta que permitirá gestionar la información de manera visual.....	134
Figura 117. Plataforma Speckle durante el proceso de creación de un nuevo modelo dentro del proyecto BIM.....	135
Figura 118. Vinculación del proyecto Speckle con el modelo BIM en Revit.	136
Figura 119. Modelo vinculado con speckle con toda la información de los parámetros... ..	137
Figura 120. Vinculación de datos del modelo BIM desde Speckle hacia Power BI ...	138
Figura 121. Extracción de los parámetros a gestionar en power BI... ..	139
Figura 122. Visualización y análisis de la información del modelo BIM en Power BI	140
Figura 123. Proceso de compilación y Hot Reload.	143
Figura 124. Aplicativo	144
Figura 125. Pantalla de Inicio de Sesión.....	145
Figura 126. Menú Principal / Dashboard.....	146
Figura 127. Escáner de Código QR Escáner de Código QR	147
Figura 128. Visualización y gestión de activos registrados.....	148
Figura 129. Registro y consulta de reportes de mantenimiento.....	26
Figura 130. Listado de reportes generados y actualización de parámetros para el modelo BIM.....	151
Figura 131. Listado de reportes generados para “Arquitectura – Base”	152
Figura 132. Modelado final de estructuras del bloque I de la IE.....	154
Figura 133. Modelado final de arquitectura del bloque I de la IE.	155
Figura 134. Modelado final de eléctricas del bloque I de la IE.	156
Figura 135. Modelado final de mobiliario del bloque I de la IE.	157
Figura 136. Panel de control para la gestión y monitoreo de activos de infraestructura	158
Figura 137. Disciplina de arquitectura.....	159
Figura 138. Modelado final de estructuras del bloque I de la IE.....	160

Figura 139. Disciplina mobiliario y equipamientos.....	161
Figura 140. Análisis de activos en estado fuera de servicio.	162
Figura 141. Resumen de los datos observados.	163

RESUMEN

El estudio tuvo como fin desarrollar un aplicativo de recolección de información integrado al BIM para la gestión del mantenimiento de la I.E Ramón Castilla y Marquesado, Jaén – 2025. La metodología fue desarrollar el aplicativo, recolección de información en campo, integrar la información con el modelo BIM y Analizar información. Entre los resultados tuvo el desarrollo y diseño del aplicativo App Mantenimiento, que mediante la interoperabilidad entre herramientas como Flutter, Firebase, Supabase, Dynamo, Revit y Power BI permitió consolidar un entorno de integración al BIM. Su implementación facilitó el registro y organización de datos de mantenimiento en campo, permitiendo identificar el estado operativo y la condición física de los activos. En el componente arquitectónico se registraron 724 elementos, de los cuales 718 se encuentran operativos y 6 fuera de servicio, con un costo estimado de mantenimiento de S/37,947.96. Las instalaciones eléctricas se identificaron 65 activos, donde 56 están operativos y 9 fuera de servicio, con un costo de S/1,440.00. Asimismo, el mobiliario y equipamiento registró 217 activos, con 207 operativos y 10 fuera de servicio, con un costo de S/12,800.00. Concluyó que el aplicativo mejora la organización de información, optimiza la gestión del mantenimiento y fortalece la toma de decisiones.

Palabras claves: Aplicativo, BIM, Gestión del mantenimiento e Institución educativa

ABSTRACT

The study aimed to develop a data collection application integrated with BIM for maintenance management at the Ramón Castilla y Marquesado Educational Institution, Jaén – 2025. The methodology consisted of developing the application, collecting field data, integrating the information with the BIM model, and analyzing the collected data. Among the results, the development and design of the App Mantenimiento application was achieved, which through interoperability between tools such as Flutter, Firebase, Supabase, Dynamo, Revit, and Power BI enabled the consolidation of an environment integrated with BIM. Its implementation facilitated the registration and organization of maintenance data in the field, allowing the identification of the operational status and physical condition of assets. In the architectural component, 724 elements were recorded, of which 718 were operational and 6 were out of service, with an estimated maintenance cost of S/ 37,947.96. In the electrical installations, 65 assets were identified, where 56 were operational and 9 were out of service, with an estimated cost of S/ 1,440.00. Likewise, furniture and equipment recorded 217 assets, with 207 operational and 10 out of service, with an estimated maintenance cost of S/ 12,800.00. It was concluded that the application improves information organization, optimizes maintenance management, and strengthens decision-making processes

Keywords: Application, BIM, Maintenance Management, Educational Institution.

1

I. INTRODUCCIÓN

1.1. Descripción de la realidad Problemática

A nivel internacional, en Inglaterra, a causa de que la industria de la construcción ha experimentado un aumento en la adopción de tecnologías digitales durante todo el ciclo de vida del proyecto, desde la fase de diseño hasta la de operación y mantenimiento se han generado grandes avances, sin embargo, algunas actividades, particularmente para las actividades de gestión de instalaciones durante la fase de operación y mantenimiento se siguen realizando de manera tradicional (Zhao, et al., 2022). En Francia, el uso de herramientas digitales para mejorar los procedimientos de mantenimiento ha aumentado en varios sectores industriales, pero es más limitado en la industria de la construcción, siendo la fase de operación y mantenimiento del ciclo de vida de un edificio la más cara (Coupry et al., 2021). En China, la fase de operación y mantenimiento de algunas estructuras complejas es desafiante, debido al hecho de que faltan tecnologías de información efectivas para facilitarla y debido a la aplicación de procedimientos de gestión tradicionales lo que genera retrasos y sobrecostos en esta fase (Xianfei, et al., 2020).

A nivel nacional, en Lima, grandes cantidades de recursos se invierte en infraestructura y equipamiento, mediante los proyectos públicos, que al terminar poseen una última fase, la operación y mantenimiento, pero el problema común es el mantenimiento en general de bienes e infraestructura, lo que puede generar que las inversiones no sean sostenibles (Hernández y Merino, 2022). En Cusco, en la actualidad se trabaja con el CAD2D en el público, solo una pequeña parte del sector privado viene ejerciendo nuevos métodos de trabajo. Ante la complejidad de los proyectos, aun se emplean métodos y herramientas que conllevan a deficientes resultados en las fases de diseño, construcción, operación y mantenimiento (Bellota, 2022). En Lima, algunas empresas presentan debilidades en su planificación, carecen de herramientas, tecnologías y métodos para el seguimiento correcto en la fase de mantenimiento de las estructuras. Deficiencias como: el personal emplea tabletas como apunte, y el CAD básico; involucrando a la planificación deficiente y su frecuencia preventiva se realizan a destiempo, generando mantenimiento correctivo no programados, demandando tiempo, costo adicional (Sánchez, 2021).

A nivel local, la infraestructura de la Instituciones Educativas enfrenta numerosos desafíos de mantenimiento, que afectan tanto la calidad del entorno educativo como la

14

seguridad de estudiantes y personal. La carencia de métodos eficientes y sistemáticos para gestionar y ejecutar las tareas de mantenimiento ha llevado a un deterioro progresivo de las instalaciones, dificultando su adecuado funcionamiento. En este contexto, la implementación de herramientas BIM (Building Information Modeling) se presenta como una solución innovadora y necesaria para optimizar el mantenimiento, permitiendo una gestión más eficiente, precisa y coordinada de los recursos y actividades, y contribuyendo así a la sostenibilidad y mejora continua de la infraestructura educativa.

Finalmente nos planteamos la siguiente interrogante: ¿Es posible mejorar la gestión del mantenimiento en la I.E. Ramón Castilla y Marquesado mediante el desarrollo de un aplicativo de recolección de información integrado al BIM, Jaén-2024?

1.2. Justificación

1.2.1. Justificación Científica

Existe una gran necesidad de desarrollar una gestión integrada para Instituciones Educativas, con el fin de garantizar su conservación y mantenimiento y permitir a los tomadores de decisiones sobre las estructuras necesitan intervenciones, mantenimiento y/o acciones de demolición. Por lo tanto, se presenta el desarrollo de un aplicativo de recolección de información para integrarlo al sistema de datos dentro de un modelo BIM. El BIM en la gestión del mantenimiento de infraestructuras educativas constituye un aporte relevante a la ciencia de la construcción y la gestión de datos. BIM no solo permite una gestión más precisa y eficiente del estado de las infraestructuras, sino que también optimiza la identificación temprana de deterioros y la planificación estratégica de intervenciones. Desde una perspectiva científica, ambas tecnologías se complementan e impulsan el desarrollo de nuevas metodologías para el mantenimiento predictivo y preventivo, contribuyendo significativamente al avance del conocimiento en los campos de la ingeniería civil, la arquitectura y la gestión de activos.

1.2.2. Justificación Económica

Desde una perspectiva económica, del desarrollo de un aplicativo de recolección de información como complemento del BIM en el mantenimiento de la infraestructura de la I.E. N°16001 Ramón Castilla Y Marquesado permite una significativa optimización de recursos.

El aplicativo trabaja con datos obtenidos a partir de una recolección preliminar de datos; por lo tanto, con estos procesos, es posible acelerar la recolección de información y la oportunidad de desarrollar bibliotecas de objeto. Las partes más importantes del aplicativo es transferir la información que se basa en el rico estudio de datos al modelado. La precisión que ofrece BIM en la planificación, gestión y ejecución de las actividades de mantenimiento no solo reduce costos, sino que minimiza errores operativos y tiempos de inactividad. Además, el uso de mantenimiento predictivo basado en BIM puede prolongar la vida útil de los edificios al identificar problemas antes de que se conviertan en reparaciones mayores y costosas. Esto contribuye directamente a una gestión más eficiente de los recursos públicos destinados a la educación, promoviendo una inversión que no solo es más efectiva, sino también sostenible a largo plazo.

1.2.3. Justificación Social

En el contexto actual, la adecuada gestión y mantenimiento de la infraestructura educativa es esencial para garantizar ambientes de aprendizaje seguros y funcionales. La creación del aplicativo y el BIM en la I.E. N°16001 Ramón Castilla y Marquesado no solo optimiza procesos de mantenimiento, sino que también promueve una administración más eficiente de los recursos y facilita la planificación a largo plazo; demostrando que es una metodología capaz de satisfacer las necesidades de los usuarios en una sociedad cada vez más desafiante. Se permitirá una mejor visualización y análisis de las condiciones de la infraestructura, lo que resulta en intervenciones más oportunas y efectivas, reduciendo costos y tiempo de respuesta ante posibles fallos. Además, este enfoque innovador contribuirá a la sostenibilidad y a la creación de un entorno educativo más adecuado para los estudiantes y el personal docente, mejorando así la calidad de la educación y el bienestar general de la comunidad educativa.

1.2.4. Justificación ambiental

La aplicación de herramientas y tecnología BIM en el mantenimiento de la infraestructura de la Institución Educativa N°16001 Ramón Castilla y Marquesado no solo contribuye a la optimización de los procesos de gestión y conservación de los recursos materiales, sino que también favorece la sostenibilidad ambiental. La implementación de BIM permite una planificación y ejecución más eficiente de las actividades de

mantenimiento, reduciendo el desperdicio de materiales y minimizando el impacto ambiental asociado con las reparaciones y renovaciones. Además, al promover una mayor precisión en la gestión de los recursos y una adecuada previsión de necesidades, se facilita la adopción de prácticas más ecológicas, como la selección de materiales de bajo impacto y la implementación de soluciones energéticamente eficientes. En consecuencia, la integración de BIM en la infraestructura educativa no solo mejora la durabilidad y funcionalidad de los espacios, sino que también respalda el compromiso con un desarrollo más sostenible y respetuoso con el medio ambiente.

1.3. Hipótesis

Mediante el desarrollo de un aplicativo de recolección de información integrado al BIM se mejora la gestión del mantenimiento en un 70% en la Institución Educativa N°16001 Ramón Castilla y Marquesado, Jaén – 2024.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar un aplicativo de recolección de información integrado al BIM para la gestión del mantenimiento de la Institución Educativa Ramón Castilla y Marquesado, Jaén – 2025

1.4.2. Objetivos específicos

- Diseñar y desarrollar un aplicativo para la recolección de información para la gestión del mantenimiento en la Institución Educativa Ramón Castilla y Marquesado.
- Implementar y evaluar el aplicativo en la recolección de datos de infraestructura para la gestión y mantenimiento en la Institución Educativa Ramón Castilla y Marquesado.
- Integrar la información recopilada del aplicativo mediante el modelo BIM para optimizar la gestión del mantenimiento en la Institución Educativa Ramón Castilla y Marquesado.

1.5. Antecedentes de la investigación

1.5.1. Internacionales

11 Barrera y Guerrero (2023) en su tesis, plantearon como objetivo realizar una propuesta de dotación y gastos de mantenimiento mediante la metodología BIM del edificio aulas oriente de la universidad francisco de paula Santander-sede Cúcuta. Como resultados obtuvieron que el consumo energético de estos equipos eléctricos es representativo, alcanzando los 183338 Kilovatios al año, lo que se traduce en un porcentaje elevado del consumo energético del edificio. Concluyendo que la metodología BIM en este proceso permitió integrar de forma eficiente la información de los diferentes elementos del edificio, lo que facilitó la detección temprana de posibles conflictos y errores en la elaboración de la propuesta de dotación y mantenimiento.

Dávila et al, (2022) en su tesis, plantearon como objetivo desarrollar una guía metodológica para la gerencia de proyectos en las etapas de planeación, monitoreo y control, para la construcción de edificaciones sostenibles integrando BIM. Como resultados obtuvieron que en el desarrollo de proyectos para que no existan reprocesos y pérdidas de tiempo, es esencial la participación activa de los Stakeholders en los cuatro grupos de procesos: iniciación, planeación, ejecución y control. Llegando a la conclusión que el aumento de la productividad en el sector de la construcción solo es posible si se cambia los paradigmas que se tienen en el sector, con el objetivo de aumentar la productividad del sector usando marcos de gerencia de proyectos y tecnología BIM.

Gómez et al, (2021) en su tesis, plantearon como objetivo realizar una caracterización de la edificación a intervenir que permita evaluar las necesidades organizar, programar y hacer seguimiento al mantenimiento preventivo ofrecido al cliente. Como resultados obtuvieron que las inversiones en mantenimiento de la Infraestructura e Instalaciones y de equipamiento al día de hoy no responden a las necesidades de las unidades, el mantenimiento preventivo ayuda a reducir los costos de realizar reparaciones a mayor escala. Llegando a la conclusión que las guías de ciclos para el mantenimiento de estructuras son herramientas eficaces para la programación de actividades, según las condiciones de la edificación.

3 Amórtegui (2021) en su tesis, estableció como objetivo realizar el análisis comparativo de la implementación del BIM en un proyecto de vivienda. Como resultados obtuvo una influencia del 2,3% sobre el costo total del proyecto y del 20,34% en el tiempo programado de ejecución, las variaciones en el alcance o cantidades de obra se midieron para el 79.365% de las actividades programadas, encontrándose diferencias entre las metodologías en el rango del -20% al 30%. Concluyó que la implementación del BIM en la planificación y fases, representa un avance a la digitalización e innovación en la construcción

4 Matarrita y Gera (2020) en su artículo científico, establecieron como objetivo plantear la metodología Building Information Modeling (BIM) en la gestión de mantenimiento hospitalario. Como resultados obtuvo que el costo de hacer modificaciones al diseño original incrementa. Además, que por cada dólar que se invierte en diseño, se tienen beneficios de \$20 en construcción y \$60 en operación y mantenimiento, el ahorro eficaz está en la optimización de la operación y mantenimiento de la infraestructura. Concluyeron que, si se gestionan las nuevas inversiones de infraestructura empleando BIM, se tendrán ahorros en el diseño y construcción, pero generalmente en la operación y mantenimiento.

1.5.2. Nacionales

Abanto (2023) en su tesis realizada en Lima, estableció como objetivo elaborar un informe de operación y mantenimiento preventivo para una estructura escolar mediante el sistema de gestión BIM. Como resultados obtuvo que la especialidad de instalaciones eléctricas es la que presenta mayores incidencias en mantenimiento, planteando efectuar el Flujo de Trabajo de Operación y Mantenimiento preventivo, BIM propone coordinación, optimización de los recursos y planificación a largo plazo los trabajos de mantenimiento. Concluyó que las plataformas de gestión de activos tienen que ser afable, de uso simple, permitiendo la designación de tareas, con avisos para recodar actividades de mantenimiento e información eficaz.

6 Zamora (2023) en su tesis desarrollada en Lima, planteó como objetivo optimizar procesos de preparación y documentaria de un proyecto para la fase de operación y mantenimiento, usando BIM. Como resultados obtuvo que, en el procedimiento de recolectar información de los especialistas, de levantamiento y seguimiento de las instalaciones ya realizadas en obra acompañadas del uso de tecnologías y el procesamiento de datos en el

modelo BIM del proyecto se redujo un 38% el tiempo de ejecución de actividades, garantizando la productividad, así como el cumplimiento de los plazos. Concluyó que el uso de esta metodología permite mejorar esta etapa final de toda infraestructura.

5 Venegas (2023) en su tesis desarrollada en Lima, planteó como objetivo determinar la relación entre la Metodología BIM y la Calidad de Servicio de Mantenimiento de una estructura Pública. Como resultados obtuvo que el BIM presenta un nivel alto y medio con 96.30% y 3.70%, la calidad de servicio tuvo un nivel alto y medio con 96,30% y 3.70%. Llegando a la conclusión que entre la metodología BIM y la calidad de servicio de mantenimiento, se tiene una relación positiva y alta con una significancia 0,001 ($<0,05$) y un coeficiente de Spearman de 0,746.

2 Sánchez (2022) en su estudio que tuvo como fin implementar una gestión de calidad utilizando la metodología BIM para mantenimiento de edificio en el distrito de Pueblo Libre, año 2021. Como resultados tuvo que en los costos se logró ahorrar un 5% del total, las metas trazadas se lograron realizar al 95% por medio del cambio en la planificación y una satisfacción del 96% de los usuarios gracias a la calidad del servicio realizado. Concluyendo que es menester realizar el cambio y utilización de nuevas metodologías que van de la mano con el desarrollo actual en los mantenimientos, como parte de información digital se utilizó el Revit 2019, un sistema integral interno de la empresa.

Hernández y Merino (2022) en su tesis realizada en Lima, plantaron como fin realizar una propuesta de implementación de BIM 7D influyente en el mantenimiento de una estructura. Como resultados obtuvieron que el presupuesto de mantenimiento para los activos del Laboratorio de Ecología y Utilización de Pastizales asciende a S/. 466.00, de la simulación se pudo determinar que hubo una reducción de 8 días respecto a los procedimientos regulares para el requerimiento de mantenimiento esta reducción da un monto de ahorros de S/. 302.00. Concluyendo que la propuesta de implementación de BIM 7D influye de manera positiva en el mantenimiento para la edificación de estudio.

Mendez (2020) en su investigación realizada en Villa El Salvador, Lima, planteó como finalidad complementar modelos BIM en programa mantenimiento de una infraestructura hospitalaria. Como resultados obtuvo que el 100% tiene un conocimiento

nulo sobre metodología BIM, el 100% califican que la comunicación es mala haciendo referencia a la subordinación del área técnica, el 50% califican que el plan de trabajo es bueno y el otro 50% opina lo contrario, que el 100% están interesados de aprender BIM y conocer sobre esta metodología generando la curiosidad de implementar este nuevo método innovador y tecnológico. Concluyó que la implementación del modelamiento basado en BIM tiene su mayor expresión aumentado la eficacia y la eficiencia en el gestionamiento del mantenimiento preventivo y correctivo.

1.5.3. Regional y/o local

Palomino et al., (2024) en su artículo científico desarrollado en Jaén, establecieron como finalidad estimar el progreso físico del proyecto mediante el uso de vehículos aéreos no tripulados (UAV) y modelado de información de construcción (BIM), la metodología utilizada se capturó 848 imágenes en alta resolución de la obra civil, se procesaron utilizando el software Agisoft 2.0.1, luego se importó a ReCap. Software Pro 2023, se utilizó el software Revit 2023, finalmente se exportó al software Power BI 2023. Como resultados se obtuvo que la obra usando la metodología propuesta fue del 42.82%, lo que no fue estadísticamente significativo comparado con el Sistema de Información de Obras Públicas (INFOBRAS) de 43,14%. Concluyendo que este método proporciona un sustituto preciso y eficaz de las técnicas tradicionales de medir el avance tangible de los proyectos.

Gonzales y Rodríguez (2021) en su tesis realizada en Cajamarca, establecieron como propósito aplicar la metodología BIM en la planificación y eficiencia de procedimientos en el área estructural de una empresa. Como resultados obtuvieron variaciones en el área de instalaciones sanitarias, eléctricas, metrados y costo y en el área de estructuras de 8.62 %, 8.33%, 36.36% y 45.26% respectivamente. El costo viable de proyecto fue de S/. 177,770.43, reduciéndose a S/. 172,273.45 el emplear BIM, logrando un ahorro de S/. 5,496.98, es decir el 3.09 %. Concluyendo que el BIM – Revit son programas muy eficaces en entidades públicas y privadas.

Yopla y Zavaleta (2021) en su investigación realizada en Cajamarca, establecieron como finalidad analizar la incompatibilidades e interferencias con la metodología BIM en la obra mercado de abastos – Los Baños del Inca. Como resultados obtuvieron que generando los modelos en REVIT se tuvo variaciones en 2.84%. Concluyeron que la metodología BIM

2

con buena planificación de trabajo y el uso de herramientas en la fase de diseño son óptimas para un proyecto.

II. MATERIAL Y MÉTODOS

2.1. Ubicación geográfica

El estudio tuvo lugar en la IE. N°16001 Ramón Castilla y Marquesado de la ciudad de Jaén del departamento de Cajamarca, cuyas coordenadas UTM WGS 84 fueron 742964.80 m E, 9369130.08 m N y altitud de 722msnm.

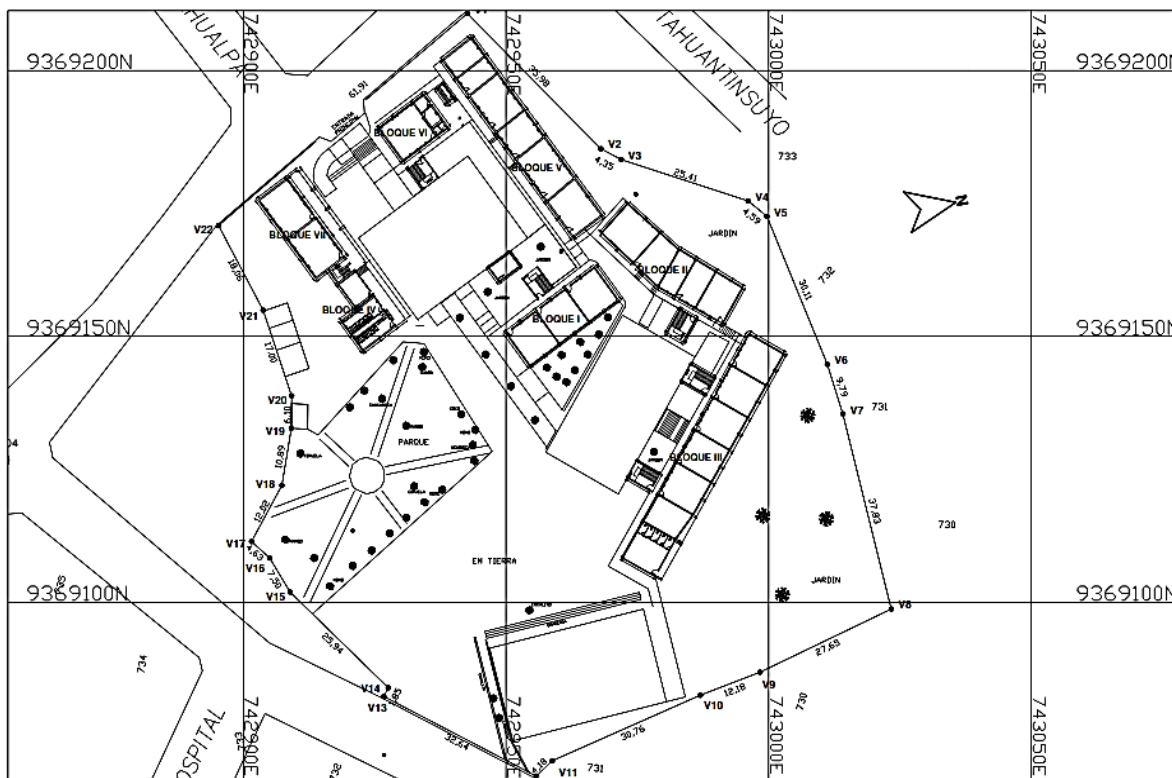
2.2. Población, muestra y muestreo

2.2.1. Población

La población de estudio para esta investigación estuvo constituida por todos los bloques de la infraestructura de la IE. N°16001 Ramón Castilla y Marquesado. Según Hernández Sampieri et al. (2014), la población se define como el conjunto completo de elementos que cumplen con ciertas características o criterios específicos para el estudio. En este contexto, la población incluyó todos los bloques de la infraestructura educativa sujetas a mantenimiento.

Figura 1

IE. N°16001 Ramón Castilla y Marquesado



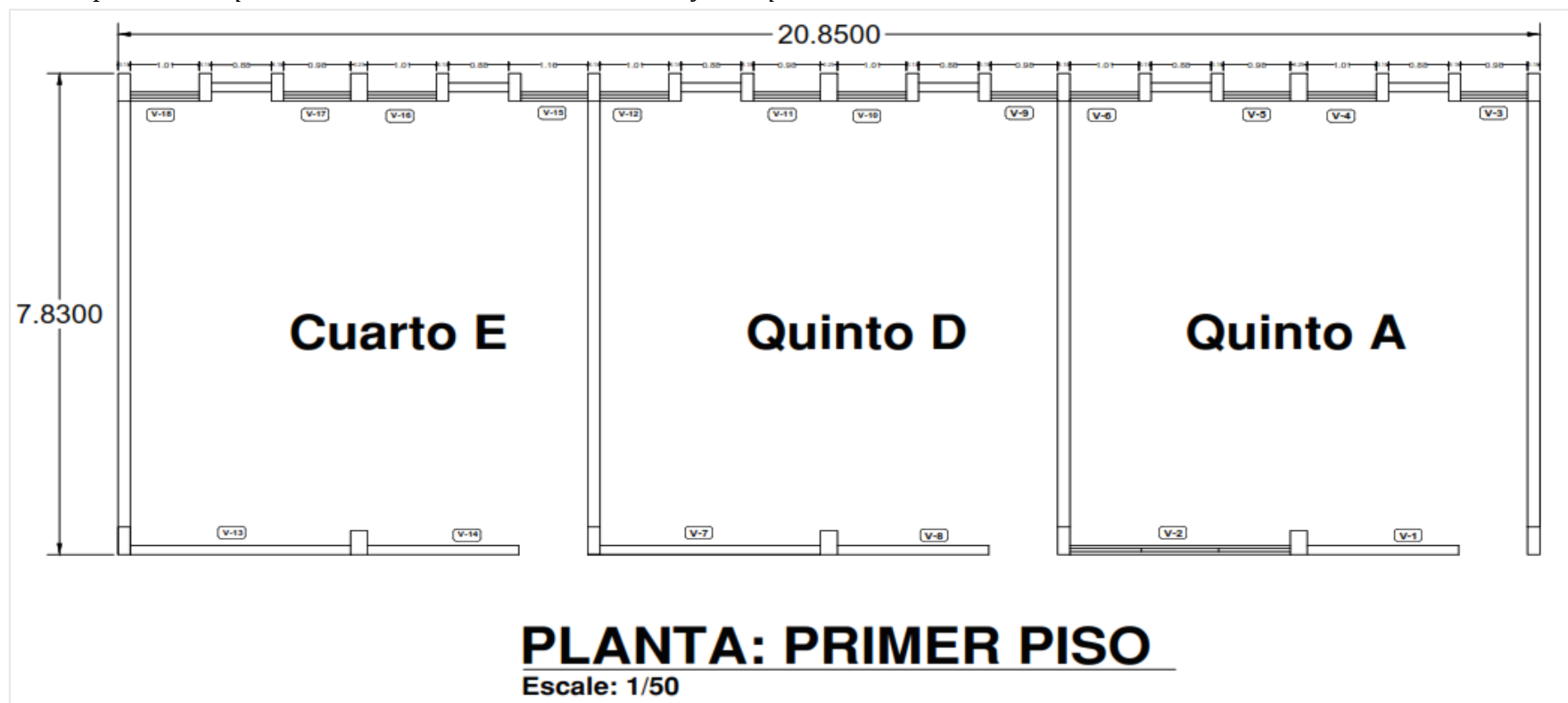
Nota. Elaboración propia.

2.2.2. Muestra

Según Arias (2016), la muestra se define como un subconjunto representativo de la población que se selecciona con el propósito de estudiar sus características. Para este estudio la muestra fue obtenida mediante el tipo no probabilístico ya que, no fue posible utilizar métodos probabilísticos por ser una investigación cualitativa. La muestra estuvo constituida por el bloque I que es de 2 niveles, conformado por 6 aulas.

Figura 2

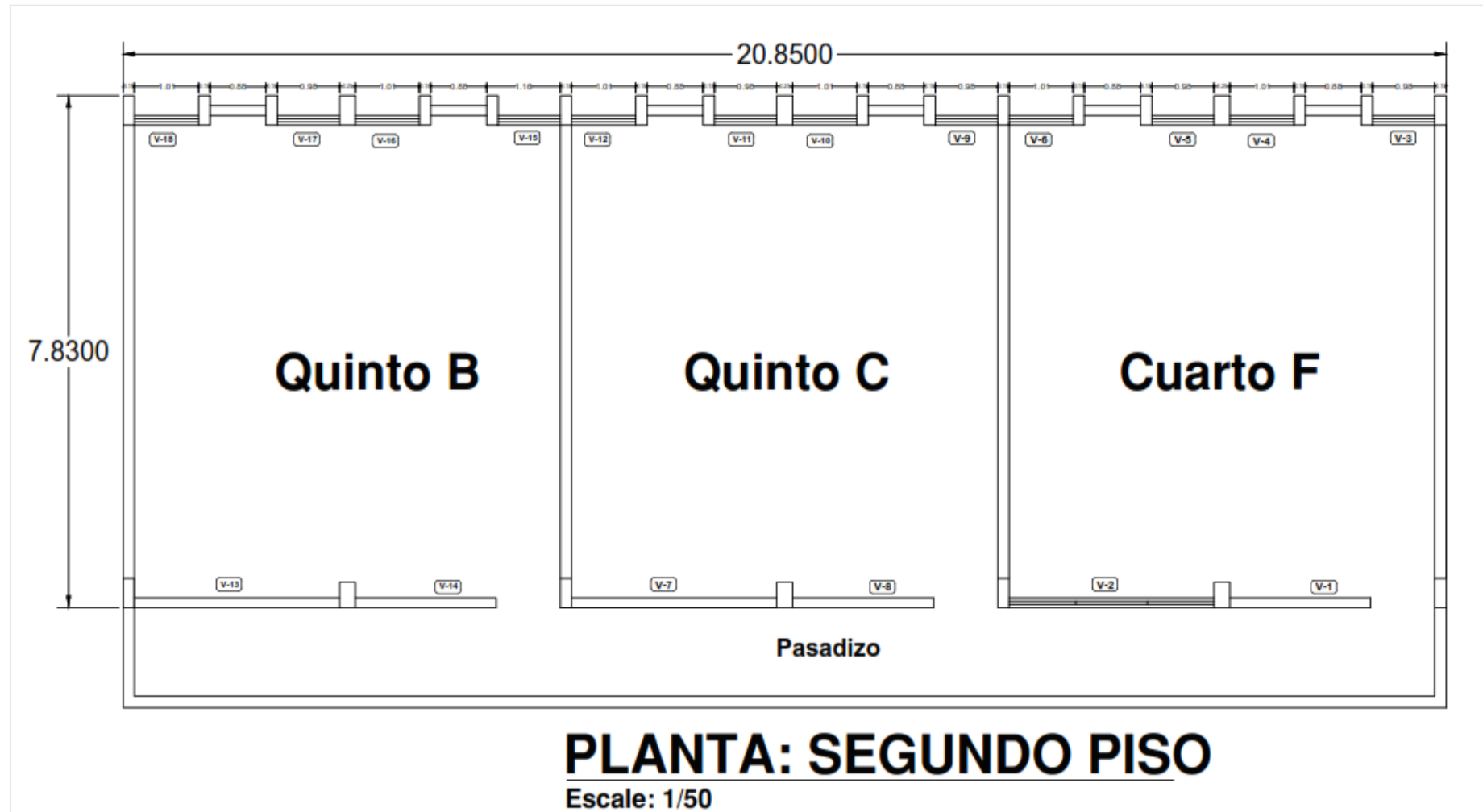
Primer piso del bloque I de la IE. N°16001 Ramón Castilla y Marquesado



Nota. Elaboración propia.

Figura 3

Segundo piso del bloque I de la IE. N°16001 Ramón Castilla y Marquesado



Nota. Elaboración propia.

2.2.3. Muestreo

La técnica de muestreo es del tipo no probabilístico por conveniencia, se realiza cuando el investigador elige la muestra bajo su criterio o juicio propio (Arias, 2016). Es por ello que el presente estudio fue carácter no probabilístico.

2.3. Metodología

2.3.1. Método

El presente estudio es de tipo básica descriptiva. Se utilizó un enfoque mixto que combino métodos cuantitativos y cualitativos. El método cuantitativo se empleó para analizar datos relacionados con el estado físico de la infraestructura mediante mediciones y métricas obtenidas a través de modelos BIM. Por otro lado, el método cualitativo permitió explorar percepciones y experiencias de los usuarios y responsables de mantenimiento respecto a la implementación de estas herramientas tecnológicas. Este enfoque mixto proporciono una comprensión integral del impacto y eficacia de las herramientas BIM en el mantenimiento de la infraestructura educativa.

2.3.2. Técnicas

Observación Directa: Se llevó a cabo una observación sistemática de la infraestructura de la institución educativa para evaluar su estado actual, las necesidades de mantenimiento y las áreas en las que se puede aplicar tecnología BIM.

Análisis Documental: Se llevó a cabo un análisis de documentos técnicos como planos arquitectónicos, informes de mantenimiento anteriores, y normativas vigentes en el sector educativo relacionadas con la infraestructura y el mantenimiento.

Encuestas a Personal Administrativo y Mantenimiento: Se aplicaron encuestas al personal administrativo y de mantenimiento de la institución educativa, con el fin de recolectar datos sobre las prácticas actuales de mantenimiento y su percepción acerca del uso de nuevas tecnologías como BIM.

2.3.3. Instrumentos

Fichas de observación estructuradas: Estas fichas incluyeron parámetros como el estado de conservación de las edificaciones, problemas recurrentes de infraestructura, y potenciales áreas de mejora mediante la implementación de tecnologías avanzadas.

Cuestionario de preguntas cerradas y abiertas: Diseñado para obtener tanto datos cuantitativos como cualitativos sobre el manejo actual de la infraestructura y la viabilidad de implementar BIM en los procesos de mantenimiento.

2.3.4. Procedimientos de recolección de datos

Para la gestión del mantenimiento se tomó en cuenta de manera integral las principales disciplinas que conforman el Bloque I, considerando los componentes arquitectónicos, las instalaciones eléctricas y el mobiliario y equipamiento, a fin de garantizar la adecuada operatividad, seguridad y conservación de la infraestructura.

2.3.4.1. FASE 1: Diagnóstico de la situación actual

Levantamiento de información del estado actual de la infraestructura e Inspección técnica de ambientes:

- Aulas
- Escalera
- Instalaciones eléctricas

Identificación de:

- Tipos de fallas recurrentes
- Estado actual
- Tiempo de respuesta
- Nivel de registro documental existente

Registro fotográfico y fichas técnicas.

2.3.4.2. FASE 2: Diseño del aplicativo de recolección de información

a. Preparación del entorno del desarrollo

El desarrollo del aplicativo, como primer paso se realizó en un entorno configurado sobre el sistema operativo Windows 11 Home de 64 bits. A continuación, se describen las herramientas y versiones utilizadas para garantizar la reproducibilidad del proyecto.

Figura 4

Versión de Flutter instalada en el sistema



```
Símbolo del sistema - flutter
C:\Users\luis_>flutter --version

A new version of Flutter is available!
To update to the latest version, run "flutter upgrade".

Flutter 3.38.6 • channel stable • https://github.com/flutter/flutter.git
Framework • revision 8b87286849 (10 weeks ago) • 2026-01-08 10:49:17 -0800
Engine • hash 6f3039bf7c3cb5306513c75092822d4d94716003 (revision 78fc3012e4) (2 months ago) • 2026-01-07 18:42:12.000Z
Tools • Dart 3.10.7 • DevTools 2.51.1

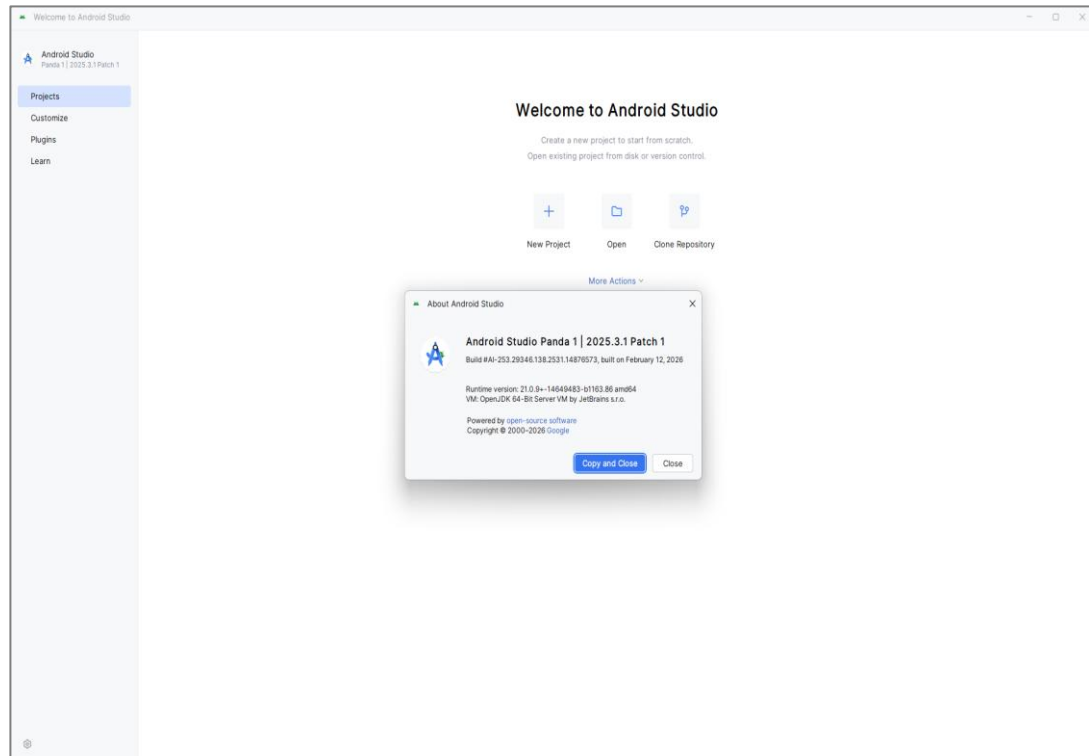
C:\Users\luis_>
C:\Users\luis_>
```

Nota. visual estudio code (2026).

La figura muestra la ejecución del comando flutter --versión en la consola del sistema Windows, mediante el cual se verifica la correcta instalación del framework Flutter utilizado para el desarrollo del aplicativo de recolección de información. Se indican que el entorno de desarrollo cuenta con Flutter versión 3.38.6 en el canal estable (stable), acompañado del lenguaje de programación Dart versión 3.10.7, así como la herramienta DevTools versión 2.51.1, utilizada para depuración y análisis del rendimiento de aplicaciones.

Figura 5

Versión de Android Studio utilizada en el desarrollo.

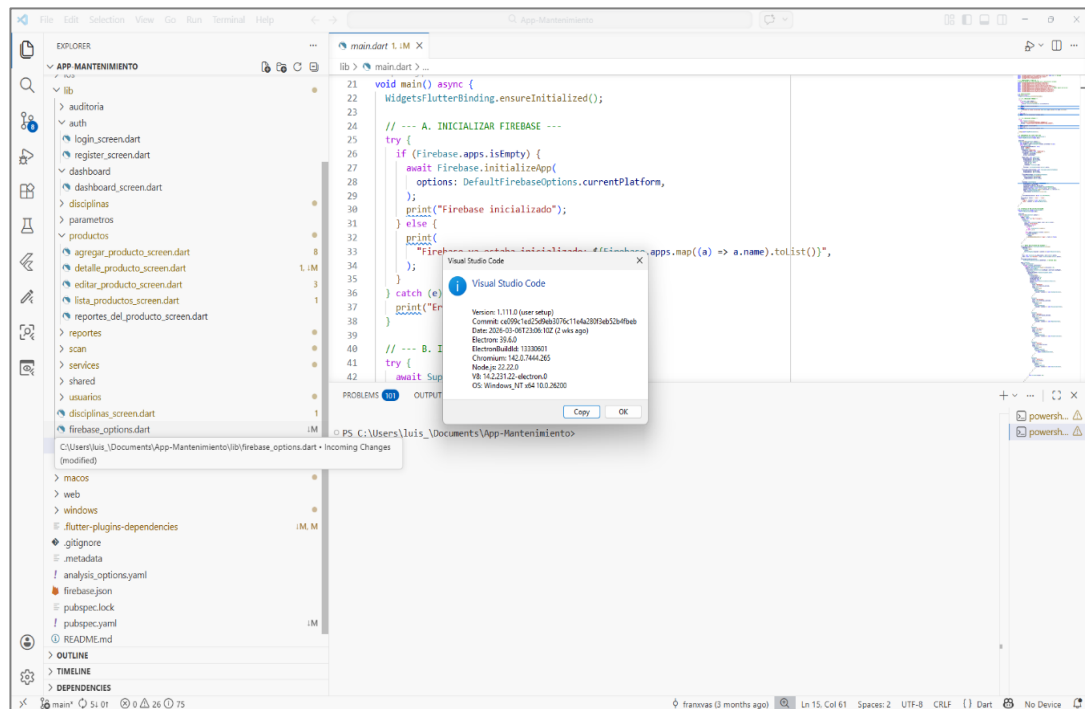


Nota. visual estudio code (2026).

La figura muestra el entorno de desarrollo Android Studio Panda | 2025.3.1 Patch 1, utilizado para la gestión del SDK de Android y la ejecución del aplicativo móvil desarrollado con Flutter. Esta herramienta permite compilar, ejecutar y realizar pruebas del aplicativo en dispositivos móviles o emuladores, garantizando su correcto funcionamiento durante el proceso de desarrollo.

Figura 6

Versión de Visual Studio Code utilizada en el desarrollo.



Nota. visual estudio code (2026).

La figura muestra el entorno de desarrollo Visual Studio Code versión 1.109.5, utilizado como plataforma principal para la programación del aplicativo. En este entorno se instalaron las extensiones Flutter y Dart, las cuales permiten desarrollar, depurar y ejecutar la aplicación móvil destinada a la recolección de información para la gestión del mantenimiento.

Navegador utilizado para pruebas web: Google Chrome. La correcta instalación y configuración de estas herramientas fue verificada mediante el comando 'flutter doctor', garantizando que el entorno se encontraba preparado para el desarrollo y ejecución del aplicativo en Android y Web.

b. Configuración de servicio en la nube

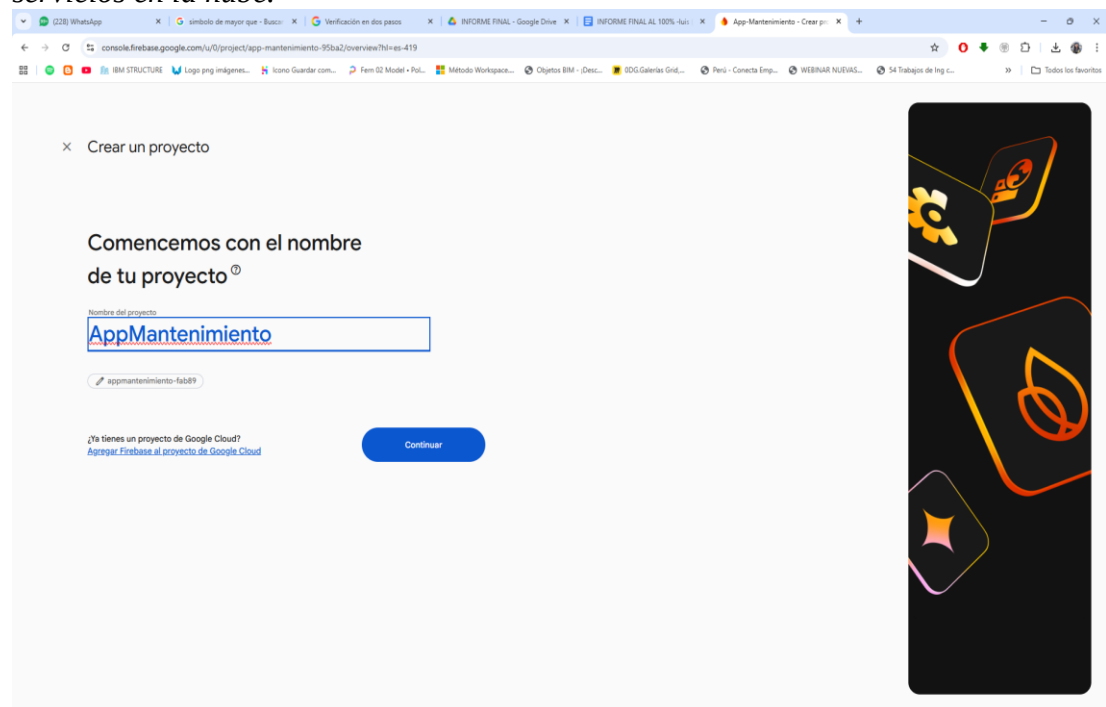
Se implementaron servicios en la nube para garantizar el almacenamiento, sincronización y disponibilidad de la información generada por el aplicativo. Estos servicios permiten la autenticación de usuarios, el almacenamiento estructurado de activos y reportes de mantenimiento, la sincronización de datos en tiempo real, la gestión de evidencias fotográficas y el acceso remoto desde múltiples dispositivos. Para ello se utilizaron Firebase,

mediante sus servicios de Authentication y Cloud Firestore para la gestión de usuarios y almacenamiento de datos, y Supabase Storage para el almacenamiento de imágenes. Esta integración permito que el aplicativo funcione tanto en dispositivos Android como en entorno web, asegurando el acceso y gestión eficiente de la información.

Seguido para la configuración de Firebase, se utilizó para: Gestión de usuarios (Authentication), almacenamiento estructurado de datos (Cloud Firestore), sincronización en tiempo real y registro de auditoría.

Figura 7

Creación del proyecto en la consola de Firebase para la configuración de los servicios en la nube.

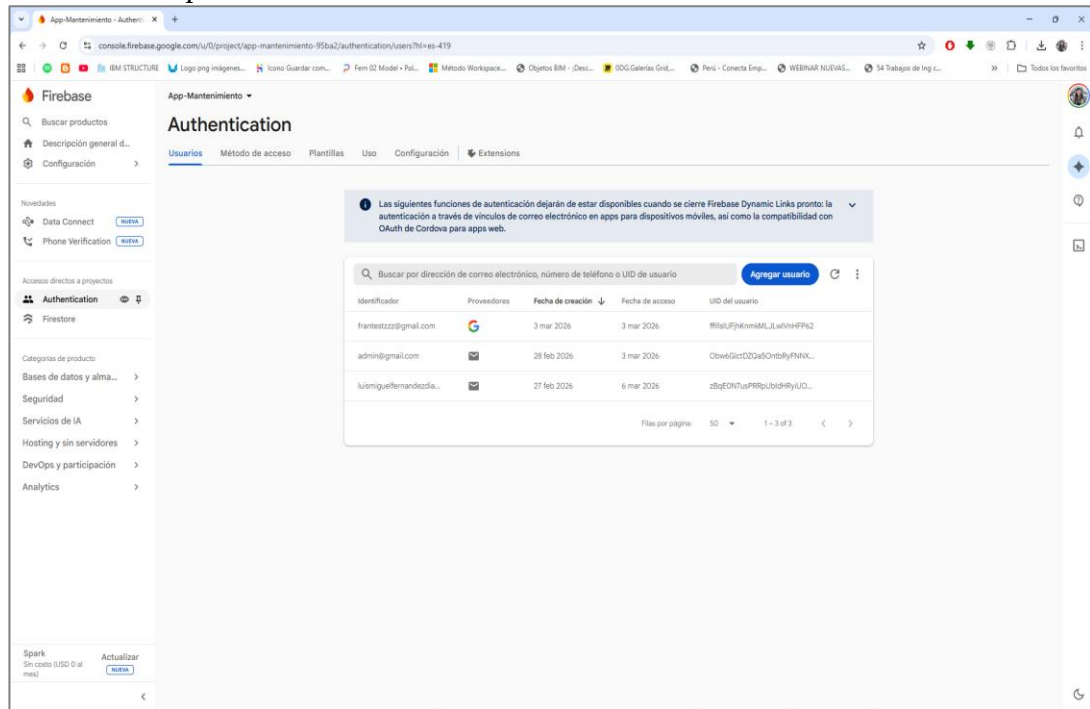


Nota. visual estudio code (2026).

La figura muestra el proceso de creación del proyecto en la consola de Firebase, donde se asigna el nombre “**AppMantenimiento**” al sistema desarrollado. Este paso permito generar el entorno en la nube necesario para integrar los servicios de autenticación, base de datos y almacenamiento que serán utilizados por el aplicativo para la gestión del mantenimiento.

Figura 8

Configuración del servicio Firebase Authentication para el control de acceso de usuarios del aplicativo.

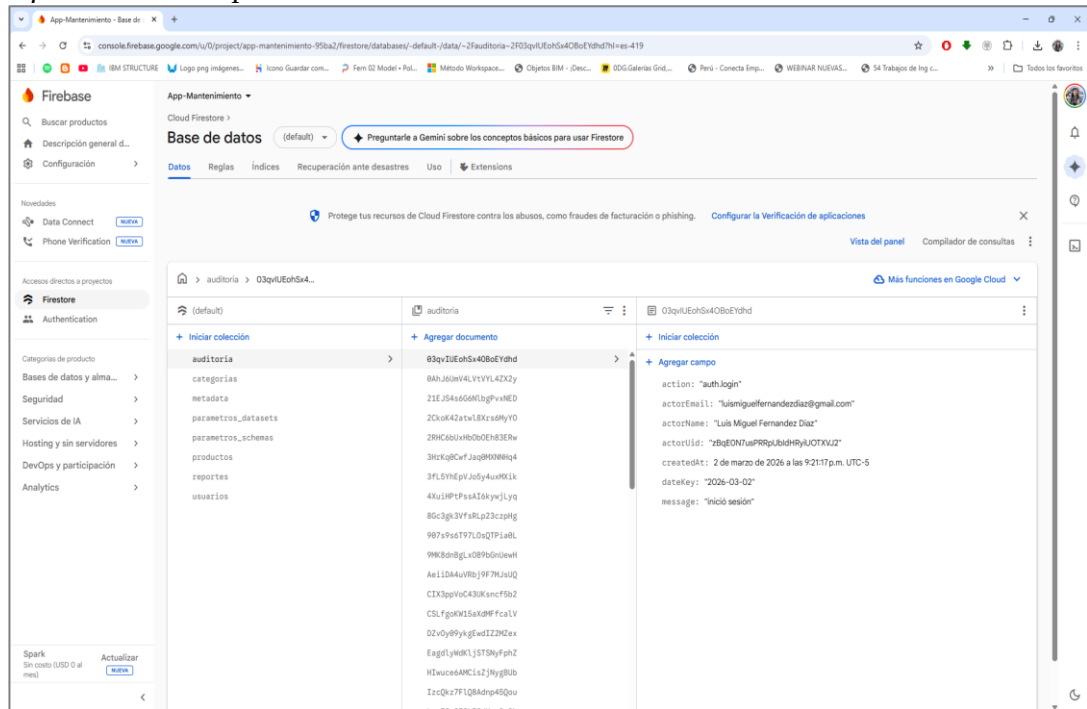


Nota. visual estudio code (2026).

La figura muestra la configuración del servicio Firebase Authentication, donde se gestionan los usuarios que tienen acceso al aplicativo. En este entorno se habilitó el método de autenticación mediante correo electrónico y contraseña, permitiendo controlar el acceso de los usuarios al sistema. Asimismo, esta configuración facilitó la administración de cuentas, la asignación de roles y el registro de actividades dentro del aplicativo para la gestión del mantenimiento.

Figura 9

Configuración de la base de datos Cloud Firestore para el almacenamiento de información del aplicativo.

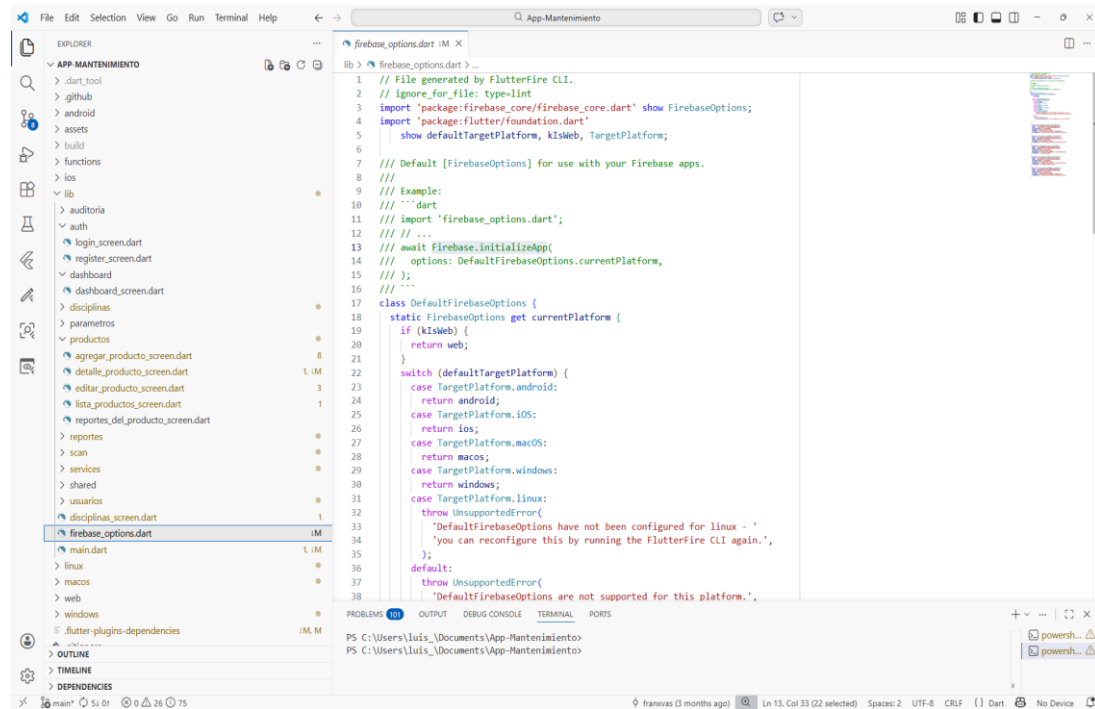


Nota. visual estudio code (2026).

La figura muestra la configuración de la base de datos Cloud Firestore en Firebase, utilizada para almacenar la información generada por el aplicativo. En esta base de datos se organizaron diferentes colecciones, tales como productos, reportes, usuarios, auditoria, categorías, parametros_schemas y parametros_datasets, las cuales permitieron gestionar de manera estructurada los activos, reportes de mantenimiento y registros de actividad del sistema. Cloud Firestore utiliza un modelo NoSQL basado en documentos, lo que facilitó la flexibilidad, escalabilidad y sincronización de los datos en tiempo real.

Figura 10

Configuración de FlutterFire para la integración del aplicativo Flutter con los servicios de Firebase.



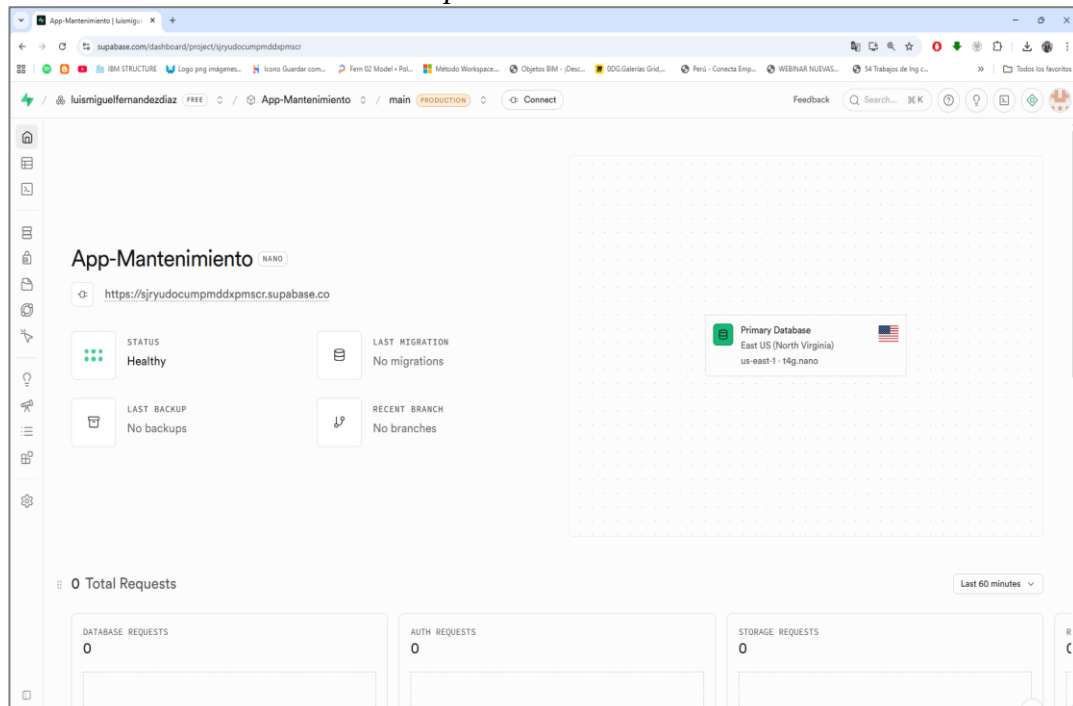
Nota. visual estudio code (2026).

La figura muestra la configuración de FlutterFire en el entorno de desarrollo Visual Studio Code, donde se generó el archivo `firebase_options.dart` mediante la ejecución del comando `flutterfire configure`. Este archivo contiene la configuración necesaria para conectar el aplicativo desarrollado en Flutter con los servicios de Firebase, permitiendo la integración con herramientas como Authentication, Cloud Firestore y otros servicios en la nube utilizados por el sistema.

Como dato: Supabase fue utilizado para el almacenamiento de imágenes generadas durante las inspecciones realizadas mediante el aplicativo. Este servicio permitió gestionar archivos multimedia en la nube de manera segura y accesible. Se optó por separar el almacenamiento de imágenes del almacenamiento estructurado de datos con el fin de mejorar la organización, eficiencia y rendimiento del sistema, utilizando Cloud Firestore para los datos del aplicativo y Supabase Storage para la gestión de evidencias fotográficas. Esta arquitectura permitió un manejo más eficiente de la información y facilita la escalabilidad del sistema.

Figura 11

Creación y configuración del proyecto en la plataforma Supabase para el almacenamiento de archivos del aplicativo.

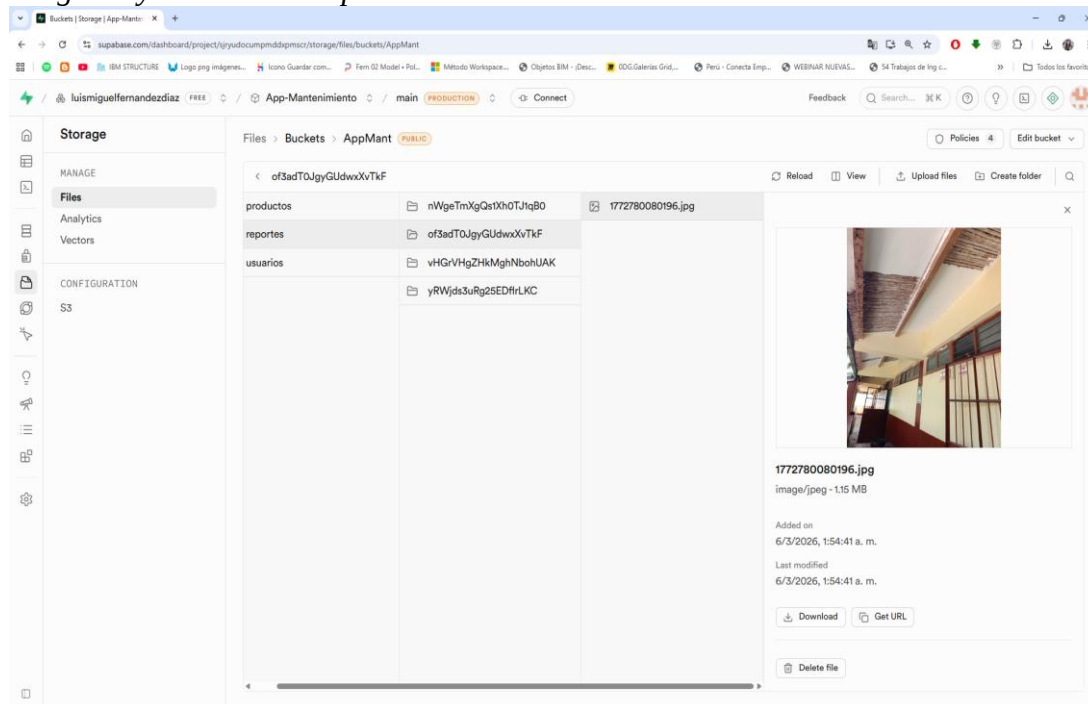


Nota. visual estudio code (2026).

La figura muestra la creación y configuración del proyecto App-Mantenimiento en la plataforma Supabase, donde se estableció el entorno en la nube para la gestión del almacenamiento (archivos). En este panel se tienen los servicios disponibles del proyecto, como Database, Auth, Storage y Realtime, los cuales permitieron administrar los recursos necesarios para el funcionamiento del aplicativo. Este entorno sirvió para gestionar el almacenamiento de imágenes de las inspecciones realizadas mediante el sistema.

Figura 12

Configuración del bucket de almacenamiento en Supabase para la gestión de imágenes y archivos del aplicativo.

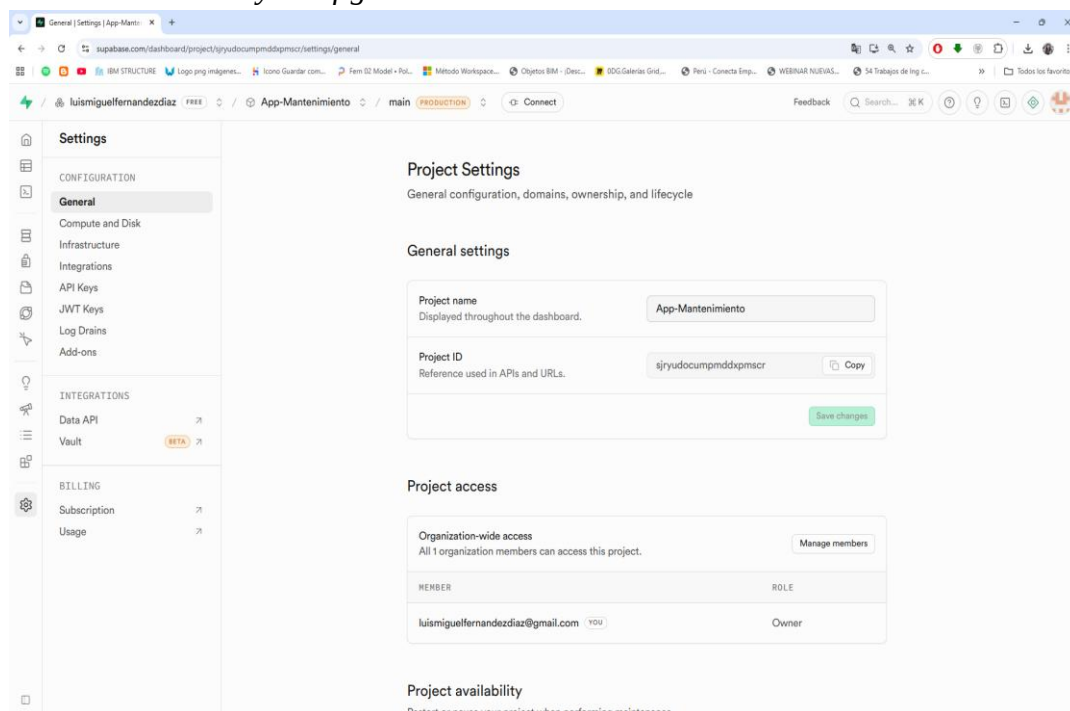


Nota. visual estudio code (2026).

La figura muestra la configuración del bucket de almacenamiento en Supabase Storage, utilizado para gestionar los archivos multimedia generados por el aplicativo. En este espacio se organizaron carpetas destinadas al almacenamiento de fotografías de activos, evidencias de fallas y archivos asociados a los reportes de mantenimiento. Este bucket permitió administrar y acceder a los archivos almacenados en la nube de manera eficiente durante las inspecciones realizadas mediante el aplicativo.

Figura 13

Obtención de URL y Configuración en Flutter



Nota. visual estudio code (2026).

La figura muestra la sección Project Settings de la plataforma Supabase, donde se visualiza la información general del proyecto App-Mantenimiento, incluyendo el Project ID, utilizado para la integración del servicio de almacenamiento con el aplicativo desarrollado en Flutter. Estos datos permitieron configurar la conexión entre el aplicativo y Supabase, facilitando la carga de imágenes generadas durante las inspecciones. Una vez cargadas, las imágenes generan una URL pública, la cual se almacenó en Cloud Firestore y vinculó al reporte correspondiente como evidencia.

La integración operativa del sistema permitió que el usuario acceda al aplicativo mediante Firebase Authentication, garantizando un acceso seguro. Una vez autenticado, el usuario consulta los activos almacenados en Cloud Firestore y genera reportes de mantenimiento. Las imágenes capturadas como evidencia se almacenaron en Supabase Storage, generando una URL pública que posteriormente se registró en Firestore dentro del reporte correspondiente. Finalmente, la información actualizada se sincronizó con el modelo BIM y es enviada a Revit mediante Dynamo. Esta configuración garantizó la disponibilidad remota de la información, sincronización en tiempo real, escalabilidad del sistema y seguridad en el acceso a los datos.

c. Estructura del proyecto

El proyecto App-Mantenimiento fue desarrollado en Flutter utilizando una arquitectura modular que facilita la escalabilidad, mantenibilidad e interoperabilidad con el entorno BIM. La organización del sistema se centraliza en la carpeta lib/, donde se agrupan los diferentes módulos que conforman el aplicativo. La estructura se diseñó siguiendo un enfoque por capas, diferenciando la capa de presentación (interfaz y pantallas), la capa de lógica de negocio (servicios y validaciones) y la capa de persistencia de datos (integración con Firebase y Supabase). Esta separación permitió mejorar la organización del código, reducir el acoplamiento entre componentes y facilitar futuras ampliaciones del sistema. Dentro del proyecto se incluyeron módulos para autenticación de usuarios, gestión de activos del modelo BIM, registro de reportes de mantenimiento, interoperabilidad con Revit y auditoría de acciones del sistema. Esta estructura permitió que la aplicación funcione como un puente entre el modelo BIM y la gestión de mantenimiento en campo, permitiendo que la información fluya desde Revit hacia la aplicación móvil y posteriormente se sincronizó nuevamente con el entorno BIM para su análisis y gestión.

Carpeta Lib/

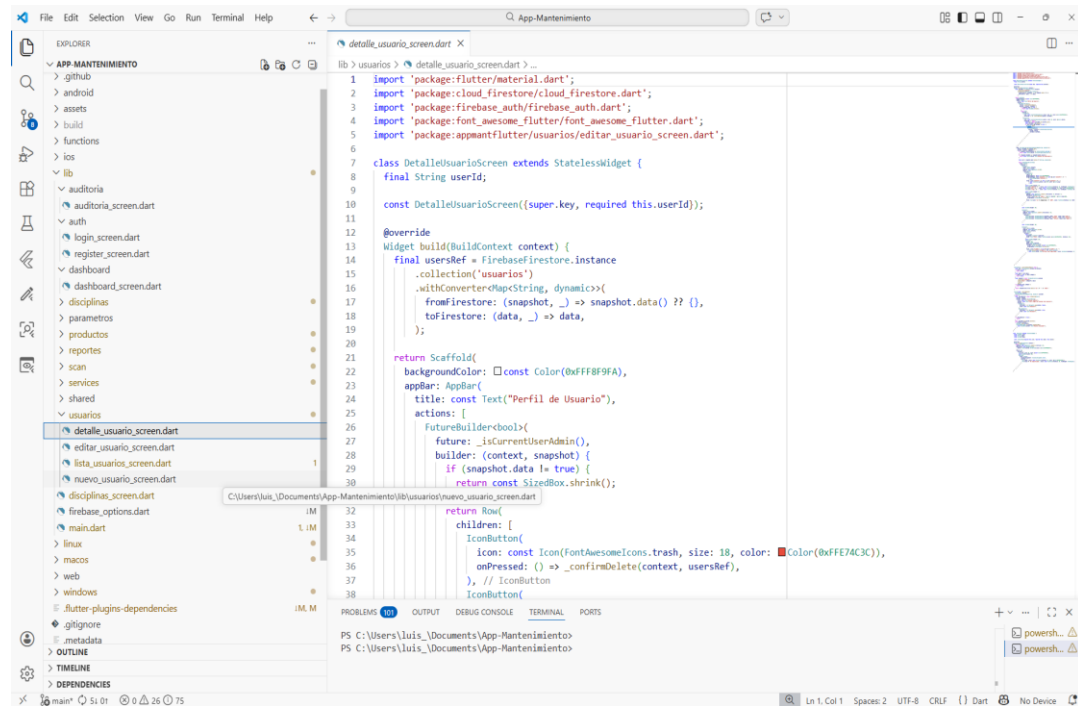
Constituyó el núcleo principal del aplicativo, ya que en ella se concentra la mayor parte del código fuente que permitió el funcionamiento del sistema. En este directorio se encuentran organizadas las pantallas, servicios, configuraciones y módulos funcionales que componen la aplicación. Con el objetivo de mantener una estructura ordenada y facilitar el mantenimiento del código, la carpeta se divide en subcarpetas especializadas, cada una con responsabilidades específicas dentro de la arquitectura del sistema. Dentro de esta organización se incluyó el módulo auth, encargado de gestionar los procesos de autenticación y registro de usuarios; el módulo dashboard, destinado a la visualización de métricas e información general del sistema; y el módulo disciplinas, utilizado para la organización técnica de los elementos según especialidades. Asimismo, el módulo parámetros permitió la interoperabilidad con el modelo BIM mediante la importación y exportación de parámetros en archivos Excel. Por otro lado, el módulo productos se encargó de la gestión de activos, mientras que el módulo reportes permitió el registro y consulta de las actividades de mantenimiento realizadas. El sistema también incorporó el módulo scan, que permitió el escaneo de códigos QR para la identificación de activos, así como el módulo services, donde se implementó la lógica de negocio del aplicativo. Finalmente, el módulo

shared agrupa los componentes reutilizables, utilizados en diferentes partes del sistema para optimizar el desarrollo y mantener la coherencia en la interfaz del aplicativo

Módulo de autenticación

Figura 14

Estructura interna del módulo de autenticación (auth/)



Nota. visual estudio code (2026).

La figura muestra la estructura interna del módulo de autenticación (auth/) del proyecto App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. En este módulo se encuentran los archivos `login_screen.dart` y `register_screen.dart`, encargados de implementar las interfaces y la lógica para el inicio de sesión y registro de usuarios dentro del aplicativo. El código desarrollado en estos archivos se integró con Firebase Authentication, permitiendo validar credenciales, gestionar el acceso al sistema y garantizando un proceso de autenticación seguro para los usuarios.

Rol del Módulo de Autenticación en la Arquitectura: La carpeta `auth/` contiene la implementación completa del sistema de autenticación del aplicativo. Este módulo es responsable de controlar el acceso al sistema mediante validación de credenciales y gestión de sesiones de usuario. Desde el punto de vista arquitectónico, este módulo:

- Actúo como puerta de entrada al sistema.
- Garantizó que solo usuarios registrados puedan acceder.
- Se integró directamente con Firebase Authentication.
- Controló el flujo de navegación hacia el dashboard principal.

El módulo fue diseñado bajo el principio de separación de responsabilidades, dividiendo la autenticación en dos pantallas independientes: inicio de sesión y registro de usuario.

Archivo `login_screen.dart`: Este archivo implementó la interfaz y lógica de inicio de sesión. Las funcionalidades principales fueron: Captura de correo electrónico, Captura de contraseña, Validación de campos obligatorios, Llamada al servicio de Firebase Authentication, Manejo de errores de autenticación y Redirección al dashboard en caso exitoso.

Archivo `register_screen.dart`: Este archivo permitió la creación de nuevos usuarios dentro del sistema. Las funcionalidades principales son: Registro mediante correo y contraseña, Validación de contraseña mínima, Confirmación de contraseña, Creación de usuario en Firebase Authentication y Registro adicional en colección usuarios de Firestore.

Integración con Auditoría: El módulo de autenticación no solo validó usuarios, sino que también registró eventos en la colección auditoria. Por ejemplo, el inicio de sesión y registro de nuevo usuario. Esto permitió mantener trazabilidad del uso del sistema.

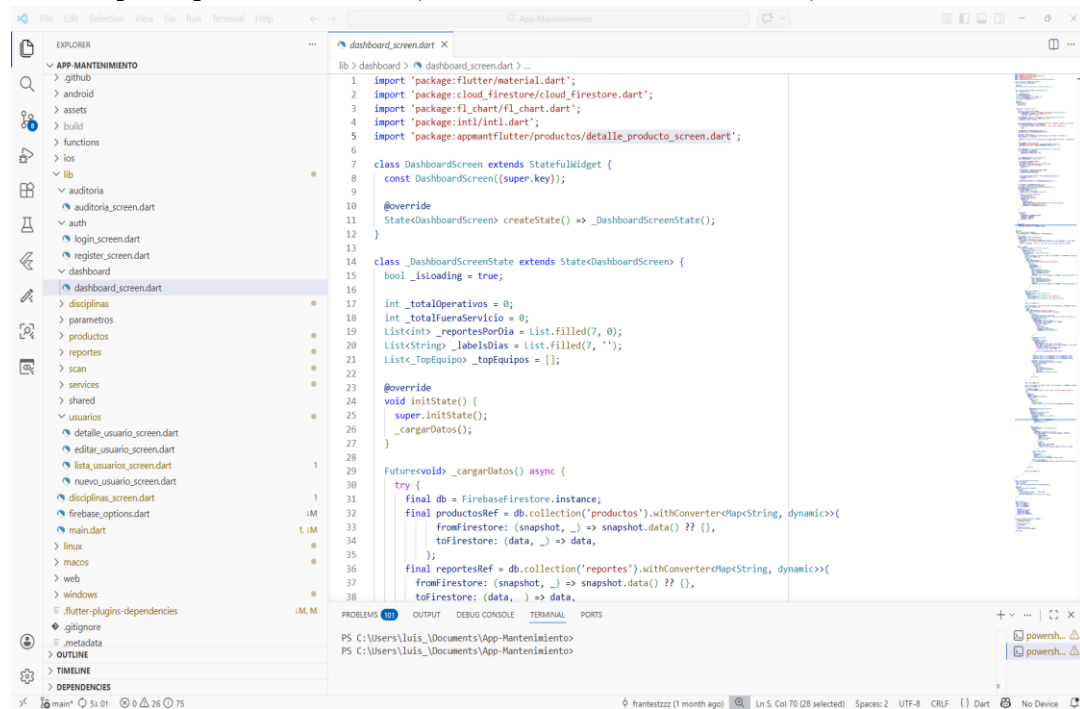
Consideraciones de Seguridad: El sistema implementó, autenticación basada en correo y contraseña, gestión automática de tokens por Firebase, control de sesión activo y restricción de acceso mediante reglas de Firestore. Esto asegura, confidencialidad, integridad y control de acceso.

Justificación de Diseño: La separación entre `login_screen.dart` y `register_screen.dart` permitió, mejor organización del código, mantenimiento sencillo, escalabilidad futura (ej. agregar recuperación de contraseña) y modularidad. Además, al centralizar la autenticación en Firebase se evitó la gestión manual de contraseñas, aumentando la seguridad del sistema.

Módulo dashboard/

Figura 15

Módulo principal – dashboard (Estructura interna del módulo)



Nota. visual estudio code (2026).

La figura muestra la estructura interna del módulo Dashboard del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. En este módulo se implementó el archivo dashboard_screen.dart, encargado de gestionar la interfaz principal del sistema donde se visualizó el estado general, métricas y datos relevantes de los activos y reportes de mantenimiento. El código estableció la conexión con Cloud Firestore para consultar información almacenada en la base de datos y presentarla de forma gráfica y resumida al usuario, permitiendo una visión general del sistema.

Rol del Dashboard en la Arquitectura del Sistema: El módulo dashboard/ representa el punto central de interacción del usuario dentro del aplicativo una vez que ha sido autenticado en el sistema. Este módulo funcionó como la interfaz principal de control, desde la cual el usuario pudo acceder a los diferentes módulos operativos de la aplicación. Desde una perspectiva arquitectónica, el dashboard actuó como un contenedor de navegación, organizando los accesos a las distintas funcionalidades del sistema y permitiendo visualizar información resumida relacionada con los activos y reportes de mantenimiento. Asimismo, este módulo gestionó la disponibilidad de determinadas funciones según el rol del usuario, facilitando

una administración adecuada de los permisos dentro del sistema. El dashboard se encontró directamente conectado con los módulos disciplinas, productos, reportes, usuarios y parámetros, permitiendo una navegación estructurada y eficiente entre los diferentes componentes del aplicativo.

Función Operativa dentro del Flujo del Sistema: Cuando el usuario ingresa sus credenciales y realiza el proceso de inicio de sesión en el aplicativo. Una vez validada correctamente la autenticación, el sistema redirigió automáticamente al módulo Dashboard, el cual actuó como la interfaz principal de operación. Desde este entorno, el usuario pudo acceder a las diferentes funcionalidades del sistema, como seleccionar la disciplina correspondiente, escanear activos mediante códigos QR, consultar el listado de productos registrados, generar reportes de mantenimiento y acceder a la auditoría del sistema según el rol asignado. De esta manera, el dashboard funcionó como un nodo de transición entre el proceso de autenticación y las operaciones principales del aplicativo, permitiendo organizar y centralizar el acceso a las distintas funciones del sistema.

Diseño de Interfaz: El diseño de la interfaz del dashboard se basó en una estructura visual que facilitó la interacción del usuario con el sistema. Generalmente, esta interfaz incorporó tarjetas (cards) organizadas por disciplina, botones de acceso rápido a las principales funcionalidades del aplicativo, así como una barra de navegación inferior o un menú lateral que permitió desplazarse entre los distintos módulos. Además, incluyó indicadores visuales de estado que brindaron información resumida sobre la condición de los activos o reportes registrados. Este diseño fue planteado con el objetivo de facilitar su uso en campo, priorizando la simplicidad en la interfaz, una navegación intuitiva y la reducción de pasos operativos, permitiendo que los usuarios registraran y consultaran información de manera rápida y eficiente durante las actividades de mantenimiento.

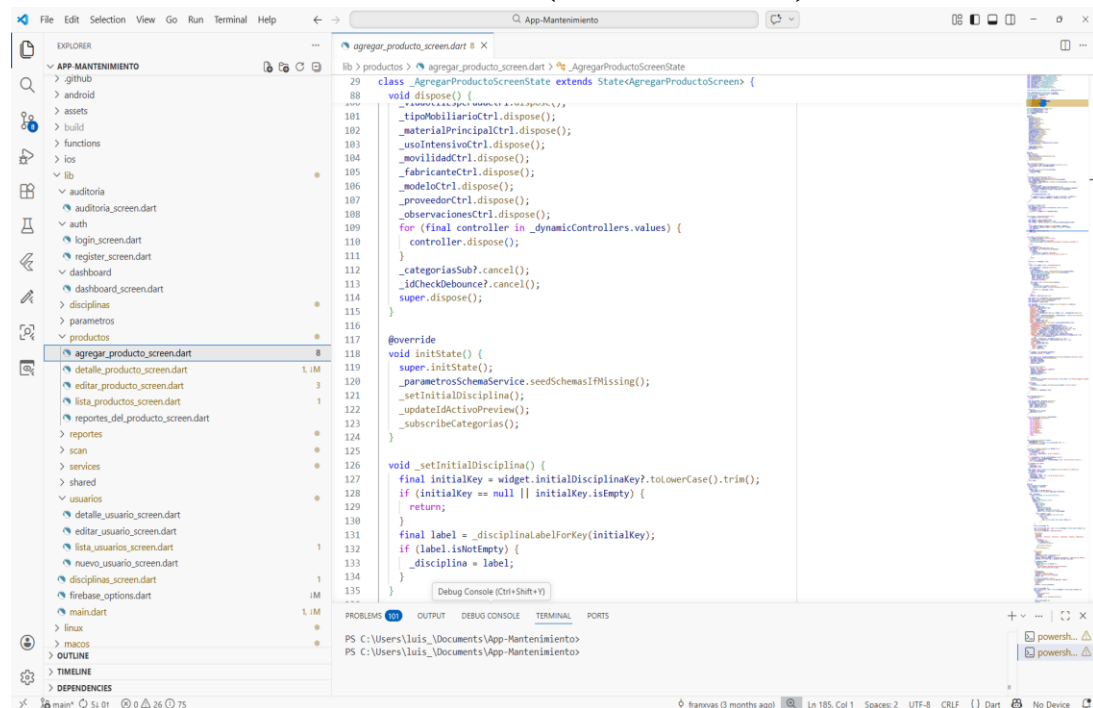
Gestión de Estado y Navegación: El módulo dashboard se encargó de gestionar la navegación principal dentro del aplicativo, utilizando mecanismos como Navigator.push o rutas nombradas para acceder a los diferentes módulos del sistema. Asimismo, permitió controlar la sesión activa del usuario, manejar el estado global de la aplicación mediante herramientas de gestión de estado y escuchar los cambios en el proceso de autenticación. La correcta implementación de este módulo permitió mantener un flujo de navegación coherente, reducir posibles errores en el desplazamiento entre pantallas y aseguró una

separación clara entre las diferentes capas del sistema, contribuyendo a una arquitectura más organizada y estable.

Módulo de gestión de activos (productos/)

Figura 16

Estructura interna del módulo Productos (Gestión de Activos)



Nota. visual estudio code (2026).

La figura muestra la estructura interna del módulo productos/ del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. En este módulo se implementaron los archivos `agregar_producto_screen.dart`, `editar_producto_screen.dart`, `detalle_producto_screen.dart` y `lista_productos_screen.dart`, los cuales permitieron gestionar los activos registrados en el sistema. A través de estos componentes se realizaron operaciones como registro, consulta, edición y visualización de información de los activos, además de la carga de imágenes asociadas mediante servicios en la nube. Este módulo constituyó el componente encargado de la gestión de activos dentro del sistema de mantenimiento.

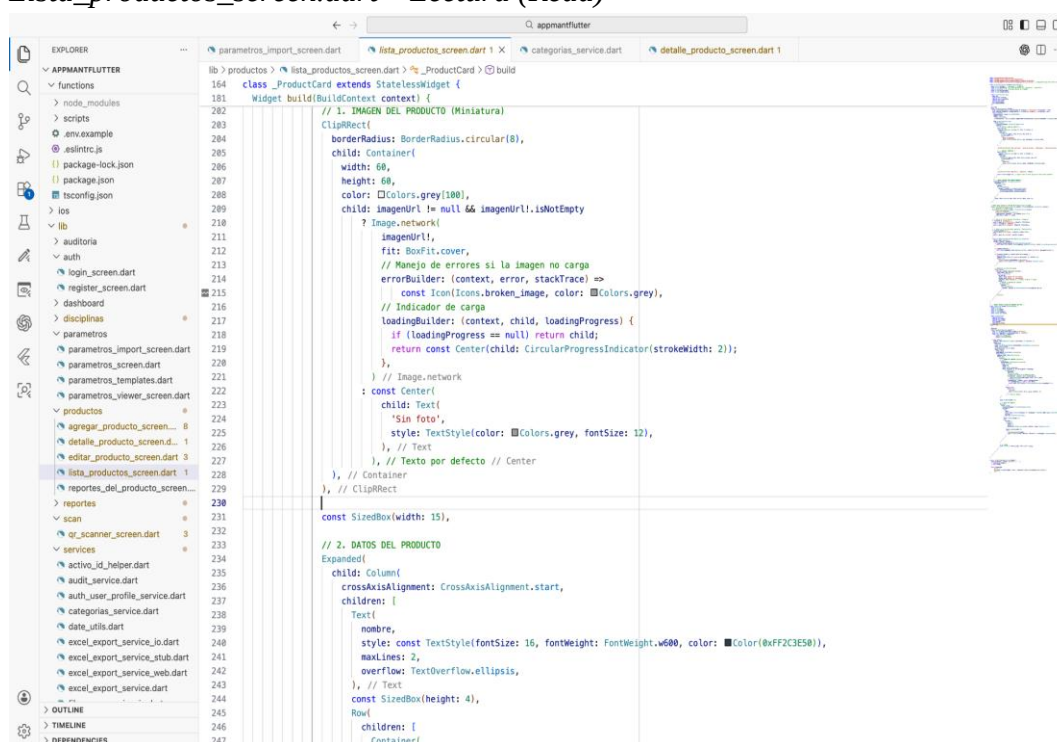
Rol del Módulo Productos en el Sistema: El módulo productos/ representó la gestión de los activos del sistema dentro del aplicativo. Cada producto correspondió a un elemento previamente modelado en Revit, cuyos parámetros fueron exportados mediante Dynamo e

incorporados posteriormente a la base de datos. Este módulo permitió visualizar, crear, editar y consultar información de los activos, además de vincular reportes de mantenimiento y gestionar su estado operativo, funcionando como un puente entre el modelo BIM y las operaciones realizadas en campo.

Estructura CRUD del Módulo: El diseño siguió una estructura clara basada en el patrón CRUD (Create, Read, Update, Delete).

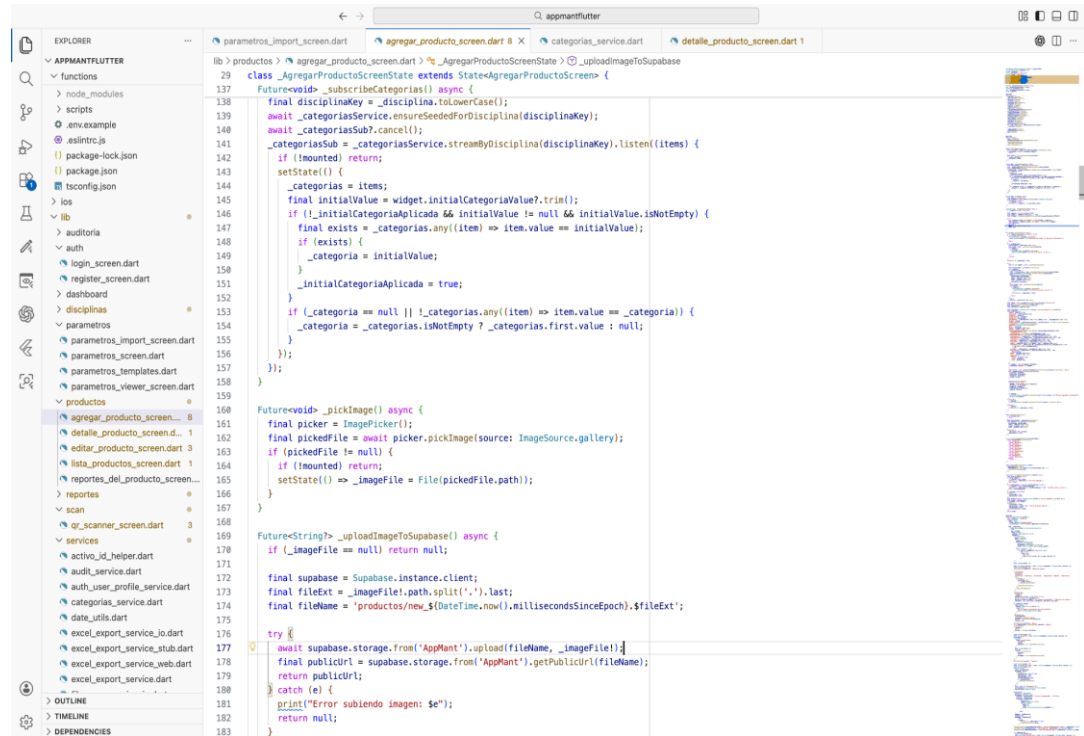
Figura 17

Lista_productos_screen.dart – Lectura (Read)



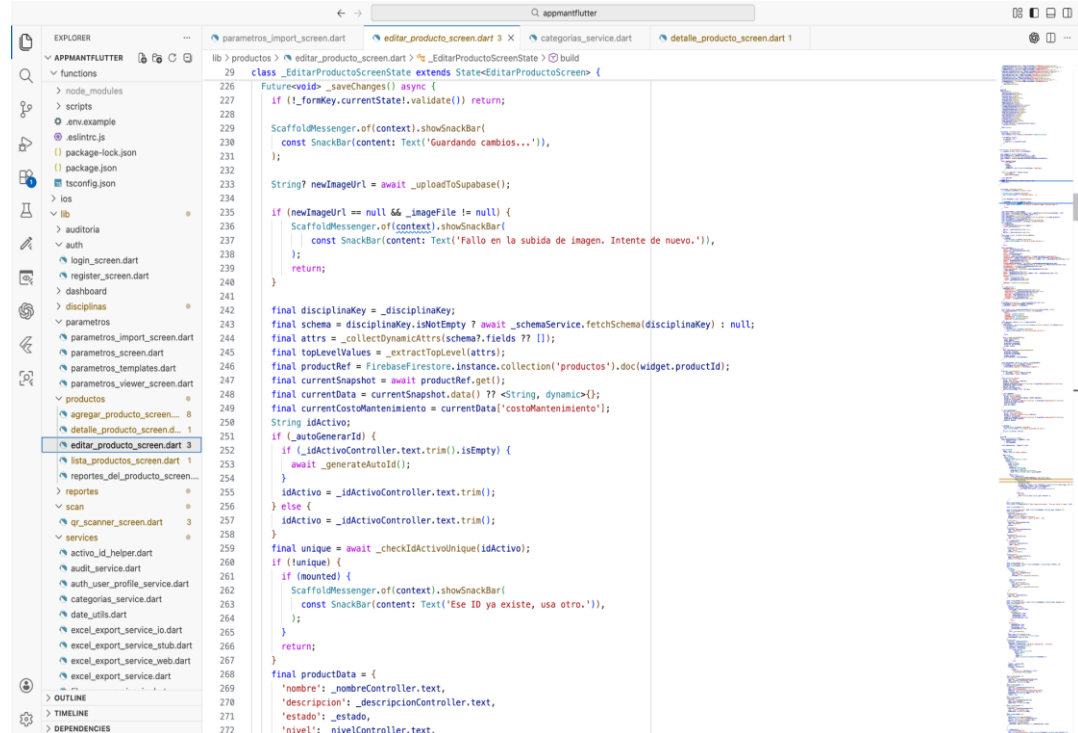
Nota. visual estudio code (2026).

La figura muestra la implementación del archivo `lista_productos_screen.dart` dentro del módulo `productos/` del aplicativo `App-Mantenimiento`, visualizada en el entorno de desarrollo `Visual Studio Code`. Este archivo se encargó de consultar la colección “productos” en la base de datos `Cloud Firestore` y mostrar el listado general de activos registrados en el sistema. A través de esta interfaz se renderizó una lista dinámica de activos, permitiendo al usuario realizar búsquedas, aplicar filtros y navegar hacia la vista de detalle de cada activo. De esta manera, este componente constituyó la vista principal para la consulta y gestión de activos dentro del aplicativo.

Figura 18**Agregar_producto_screen.dart – Creación (Create)**

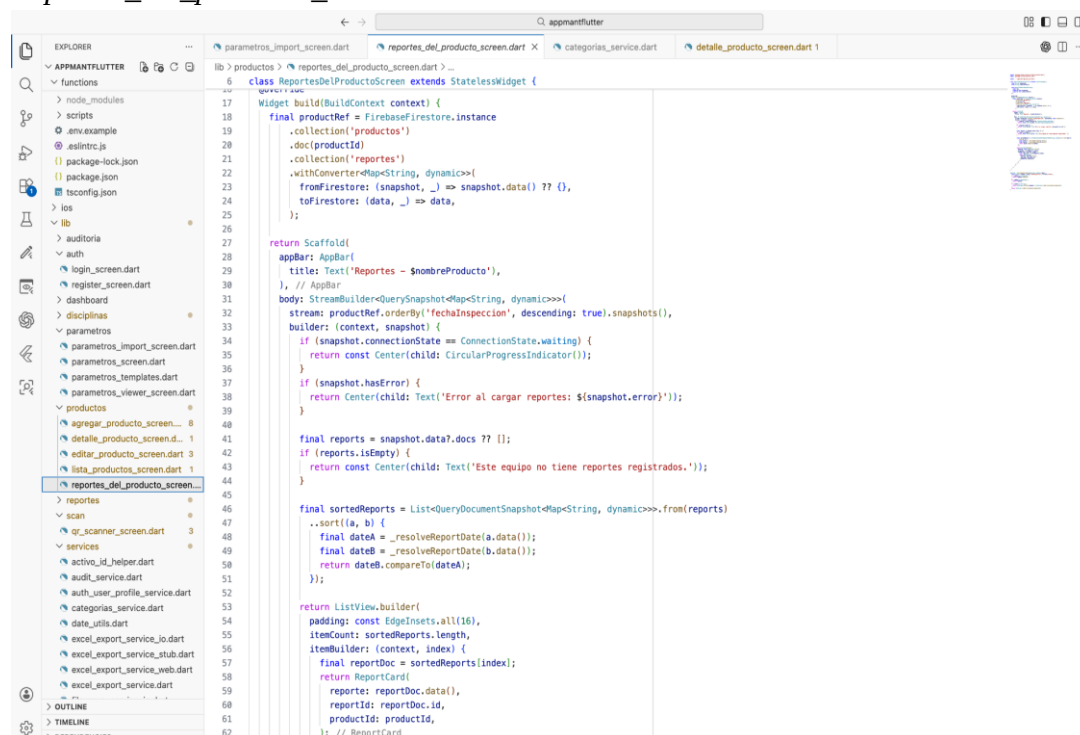
Nota. visual estudio code (2026).

La figura muestra la implementación del archivo `agregar_producto_screen.dart` dentro del módulo `productos/` del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este archivo permitió registrar nuevos activos en el sistema, implementando formularios para el ingreso de información técnica del elemento. Asimismo, el código incorporó el uso de `ImagePicker` para seleccionar imágenes del dispositivo, el método `_pickImage()` para capturar la imagen, el método `_uploadImageToSupabase()` para subirla al almacenamiento en la nube y el método `_guardarProducto()` para registrar la información del activo en Cloud Firestore. Durante este proceso, la imagen seleccionada se almacenó en Supabase Storage, generándose una URL pública que posteriormente fue guardada en el documento correspondiente dentro de Firestore. Esta implementación evidenció una integración híbrida, donde Firestore gestionó los datos estructurados del activo y Supabase administró los archivos multimedia asociados.

Figura 19***Editar_producto_screen.dart – Actualización (Update)***

Nota. visual estudio code (2026).

La figura muestra la implementación del archivo `editar_producto_screen.dart` dentro del módulo `productos/` del aplicativo `App-Mantenimiento`, visualizada en el entorno de desarrollo Visual Studio Code. Este archivo permitió modificar la información de los activos registrados en el sistema mediante formularios que cargaron los datos existentes y permitieron su actualización. A través de esta interfaz fue posible cambiar el estado del activo, actualizar parámetros técnicos, modificar el nivel de criticidad y actualizar la ubicación u otras características del elemento. Estas modificaciones fueron almacenadas posteriormente en Cloud Firestore, lo que permitió mantener la información del modelo digital actualizada y sincronizada con la realidad operativa registrada en campo.

Figura 20**Reportes_del_producto_screen.dart – Relación Uno a Muchos**

Nota. visual estudio code (2026).

La figura muestra la implementación del archivo reportes_del_producto_screen.dart dentro del módulo productos/ del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este archivo permitió gestionar la relación uno a muchos entre un activo (producto) y sus reportes de mantenimiento registrados en el sistema. A través de esta interfaz fue posible consultar los reportes asociados a un activo, visualizar el historial de mantenimiento, analizar la frecuencia de fallas y acceder a las evidencias fotográficas registradas durante las inspecciones. Esta implementación permitió organizar la información de manera estructurada en Cloud Firestore, facilitando la gestión y seguimiento del mantenimiento de cada activo.

Figura 21*Integración con Supabase (Evidencias)*

```
169 Future<String?> _uploadImageToSupabase() async {
170     if (_imageFile == null) return null;
171
172     final supabase = Supabase.instance.client;
173     final fileExt = _imageFile!.path.split('.').last;
174     final fileName = 'productos/new_${DateTime.now().millisecondsSinceEpoch}.$fileExt';
175
176     try {
177         await supabase.storage.from('AppMant').upload(fileName, _imageFile!);
178         final publicUrl = supabase.storage.from('AppMant').getPublicUrl(fileName);
179         return publicUrl;
180     } catch (e) {
181         print("Error subiendo imagen: $e");
182         return null;
183     }
184 }
185
```

Nota. visual estudio code (2026).

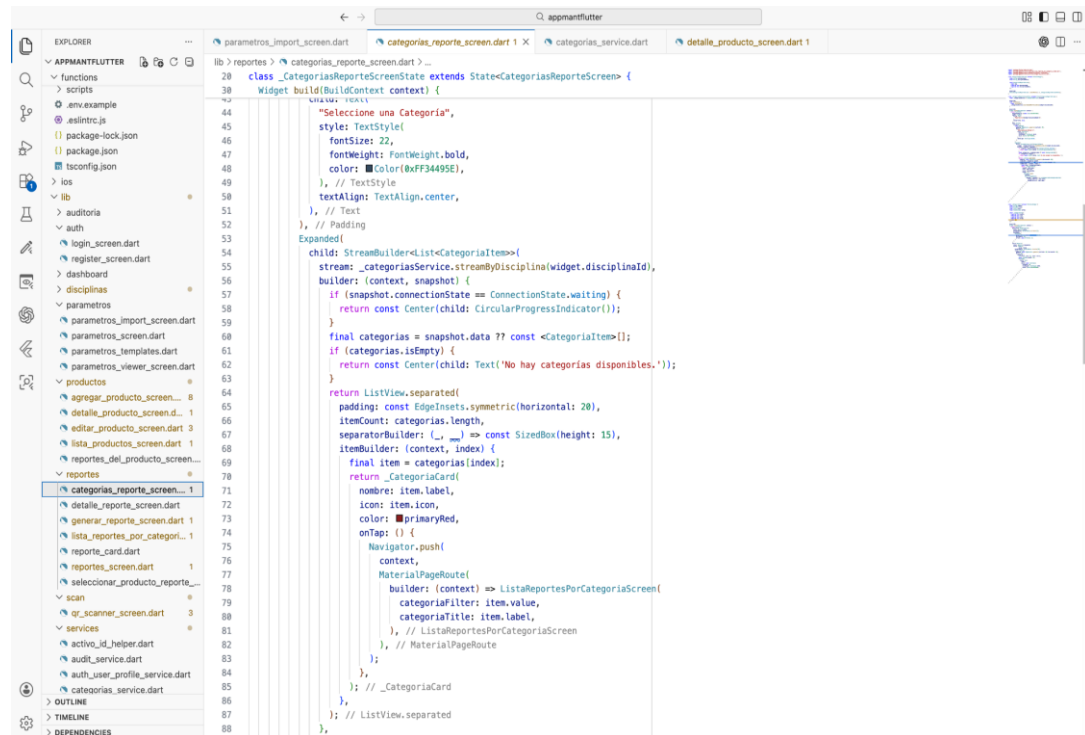
La figura muestra la implementación del método `_uploadImageToSupabase()` dentro del código del aplicativo App-Mantenimiento, donde se realizó el proceso de carga de imágenes asociadas a los activos. En este procedimiento, el sistema generó un nombre único para el archivo, posteriormente subió la imagen al bucket de almacenamiento en Supabase y obtuvo una URL pública que permitió acceder al archivo almacenado en la nube. Esta URL fue registrada posteriormente en Cloud Firestore, vinculándose con el documento correspondiente del activo o reporte. Este proceso permitió mantener una separación funcional entre los datos estructurados y los archivos multimedia, donde Firestore gestionó la información del sistema y Supabase administró las evidencias visuales, garantizando una arquitectura eficiente y organizada para la gestión de la información.

Flujo Operativo del Módulo Productos: Inició cuando el activo fue modelado en el entorno BIM mediante Revit. Posteriormente, los parámetros del elemento fueron exportados utilizando Dynamo y almacenados en la base de datos Cloud Firestore. A partir de esta información, el aplicativo consultó los activos registrados y permitió que el usuario generara reportes de mantenimiento asociados a cada elemento. Durante este proceso, el estado del activo pudo ser actualizado según las condiciones observadas en campo, y la información generada pudo sincronizarse nuevamente con el modelo BIM en Revit. De esta manera, este módulo constituyó el núcleo de la interoperabilidad entre el entorno BIM y el aplicativo de gestión de mantenimiento.

Módulo de reportes (reportes/)

Figura 22

Estructura interna del módulo Reportes.



Nota. visual estudio code (2026).

La figura muestra la estructura interna del módulo reportes/ del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. En este módulo se implementaron archivos como `categorias_reporte_screen.dart`, `generar_reporte_screen.dart`, `lista_reportes_por_categoria_screen.dart`, `reporte_card.dart` y `reportes_screen.dart`, los cuales permitieron gestionar el registro, consulta y organización de los reportes de mantenimiento asociados a los activos. A través de estos componentes se pudieron clasificar los reportes por categorías, generar nuevos registros de mantenimiento, visualizar listados de reportes y mostrar información resumida mediante tarjetas de reporte. Este módulo constituyó el componente encargado de documentar las incidencias, registrar intervenciones de mantenimiento y realizar el seguimiento del estado de los activos dentro del sistema.

Rol del Módulo Reportes en el Sistema: El módulo reportes/ implementó el proceso de registro, clasificación, consulta y análisis de incidencias de mantenimiento asociadas a los activos modelados. A través de este módulo fue posible generar reportes de fallas, clasificar

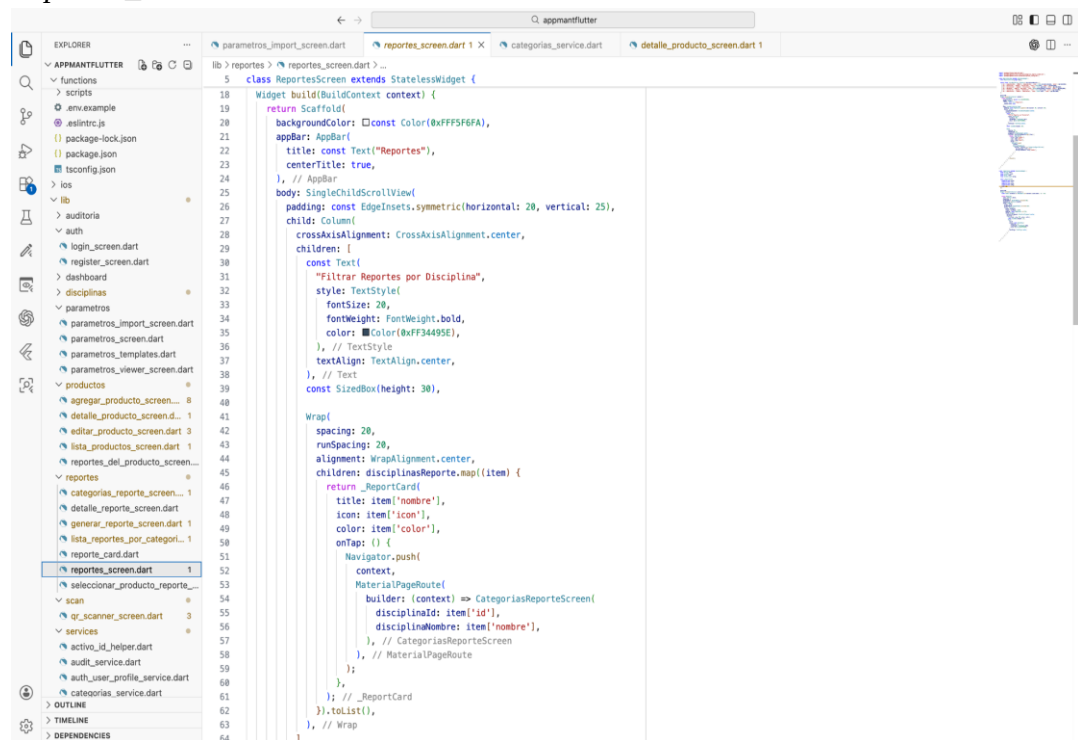
los reportes según disciplina y categoría, asociarlos a un activo específico, registrar el estado anterior y el nuevo estado del elemento, así como adjuntar evidencias y consultar el historial de intervenciones realizadas. Desde el punto de vista arquitectónico, este módulo representó el núcleo operativo del sistema de mantenimiento, ya que permitió gestionar y documentar las incidencias detectadas durante las actividades realizadas.

Flujo General de Generación de Reporte: El flujo implementado en este módulo inició con la selección de la disciplina correspondiente, seguida de la selección de la categoría del reporte y del activo asociado. Posteriormente, el usuario registró la descripción de la incidencia y la condición detectada, además de adjuntar una imagen como evidencia del evento registrado. Una vez completada esta información, los datos fueron almacenados en Cloud Firestore, realizándose además la actualización del estado del activo correspondiente. Este diseño permitió mantener coherencia en el proceso operativo y garantizar la trazabilidad de las incidencias registradas en el sistema de mantenimiento.

Análisis de Archivos Internos: Tenemos lo siguiente:

Figura 23

Reportes_screen.dart.

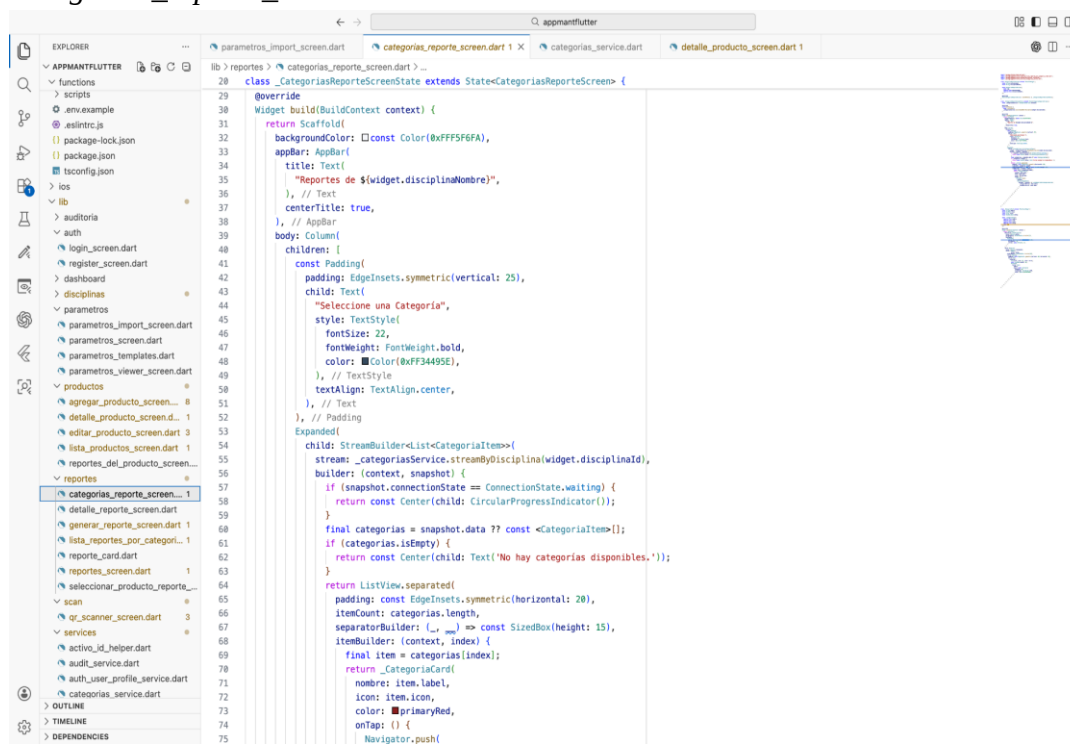


Nota. visual estudio code (2026).

La figura muestra la implementación del archivo reportes_screen.dart dentro del módulo reportes/ del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este archivo actuó como la pantalla principal del sistema de reportes, funcionando como el punto de entrada al módulo de gestión de incidencias de mantenimiento. A través de esta interfaz se organizó la navegación hacia las diferentes categorías de reportes, permitiendo al usuario acceder a las opciones de registro, consulta y clasificación de incidencias. Asimismo, esta pantalla permitió visualizar de forma general las incidencias registradas, actuando como el contenedor principal del flujo operativo del módulo de reportes.

Figura 24

Categorias_reporte_screen.dart



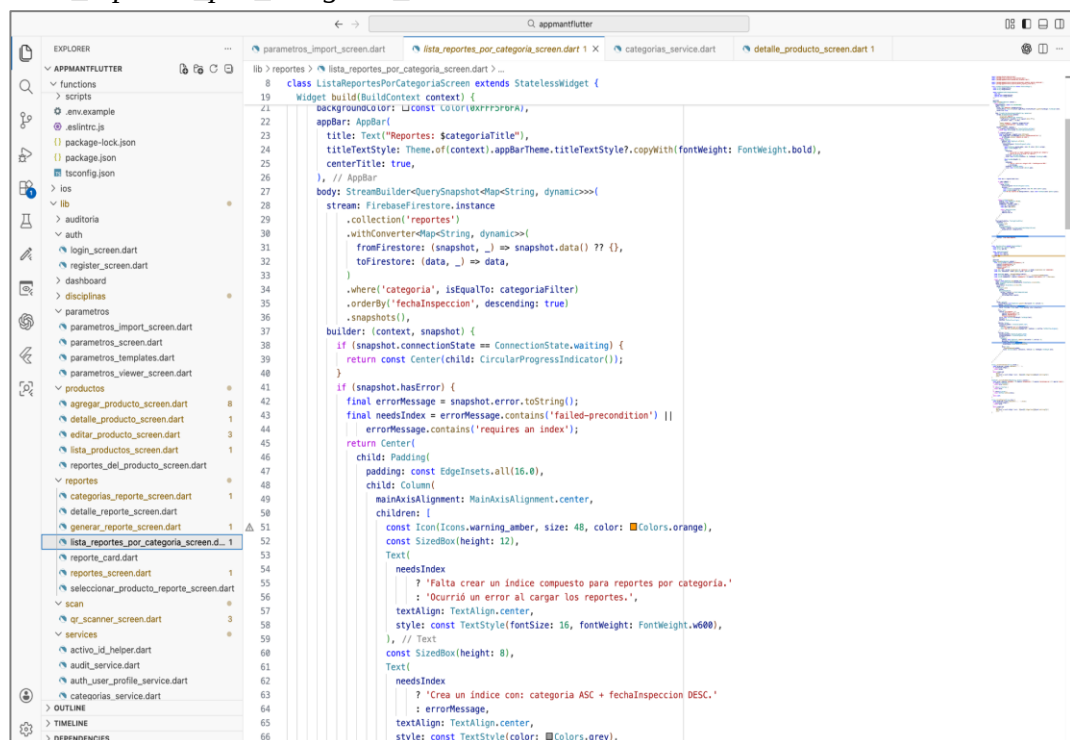
Nota. visual estudio code (2026).

La figura muestra la implementación del archivo categorias_reporte_screen.dart dentro del módulo reportes/ del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. En este archivo se definió variables como disciplinaId y disciplinaNombre, las cuales permitieron organizar los reportes según la disciplina técnica a la que pertenece el activo. Asimismo, el código utilizó el servicio CategoriasService.instance, lo que evidenció una separación entre la lógica de acceso a datos y la interfaz de usuario, siguiendo buenas prácticas de arquitectura de software. A través de

este componente, el sistema carga las categorías de reporte asociadas a una disciplina específica, garantizó la existencia de categorías base y permitió al usuario seleccionar la categoría correspondiente para continuar con el flujo de registro de incidencias. Este diseño facilitó una clasificación estructurada y escalable de los reportes dentro del sistema de mantenimiento.

Figura 25

Lista_reportes_por_categoria_screen.dart



Nota. visual estudio code (2026).

La figura muestra la implementación del archivo *lista_reportes_por_categoria_screen.dart* dentro del módulo reportes/ del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este archivo permitió consultar y visualizar los reportes de mantenimiento filtrados según una categoría específica, realizando una consulta a la base de datos Cloud Firestore. A través de esta interfaz se renderizó dinámicamente el listado de reportes asociados a la categoría seleccionada, permitiendo al usuario analizar las incidencias registradas y navegar hacia la vista de detalle de cada reporte. Esta funcionalidad facilitó un análisis sectorizado de las fallas o intervenciones de mantenimiento, mejorando la organización y consulta de la información dentro del sistema.

Figura 26*Generación_reporte_screen.dart*

```

lib > reportes > generar_reporte_screen.dart > _GenerarReporteScreenState > @_buildReadOnlyField
32 class _GenerarReporteScreenState extends State<GenerarReporteScreen> {
115 Future<void> _loadDisciplina() async {
127   final disciplina = productDoc.data()?['disciplina']?.toString().toLowerCase() ?? '';
128   if (mounted) {
129     setState(() {
130       _disciplinaKey = disciplina;
131       _tipoReporte = null;
132       _tipoMantenimiento = null;
133       _requiereReemplazo = false;
134     });
135   }
136 }
137
138 Future<int> _getAndIncrementReportCounter() async {
139   final counterRef = FirebaseFirestore.instance
140     .collection('metadatos')
141     .withConverter<ShapeString, dynamic>({
142       fromFirestore: (snapshot, _) => snapshot.data() ?? {},
143       toFirestore: (data, _) => data,
144     })
145     .doc('counters');
146
147   return FirebaseFirestore.instance.runTransaction(transaction) async {
148     final snapshot = await transaction.get(counterRef);
149     int currentNumber;
150
151     if (!snapshot.exists || snapshot.data()?['report_nro'] == null) {
152       currentNumber = 1;
153     } else {
154       currentNumber = (snapshot.data()?['report_nro'] as int) + 1;
155     }
156
157     transaction.set(counterRef, {'report_nro': currentNumber});
158     return currentNumber;
159   });
160 }
161
162 Future<void> _saveReport() async {
163   if (!_isSaving) return;
164   if (!_formKey.currentState!.validate()) return;
165   setState(() {
166     _isSaving = true;
167   });
168   if (_requiresTipoReporte && _tipoReporte == null) {
169     if (mounted) {
170       ScaffoldMessenger.of(context).showSnackBar(
171

```

Nota. visual estudio code (2026).

La figura muestra la implementación del archivo generar_reporte_screen.dart dentro del módulo reportes/ del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este archivo constituyó el componente principal para la creación de reportes de mantenimiento, ya que implementó el formulario utilizado para el registro de incidencias detectadas en los activos. A través de esta interfaz se capturaron y validaron los datos del reporte, registrándose información como acción recomendada, nombre del activo, categoría, comentarios, condición física, estado anterior, estado nuevo, disciplina, fecha e impacto de la incidencia. Una vez completado el formulario, la información fue almacenada en la colección “reportes” de Cloud Firestore.

17

Figura 27

Seleccion_producto_reporte_screen.dart

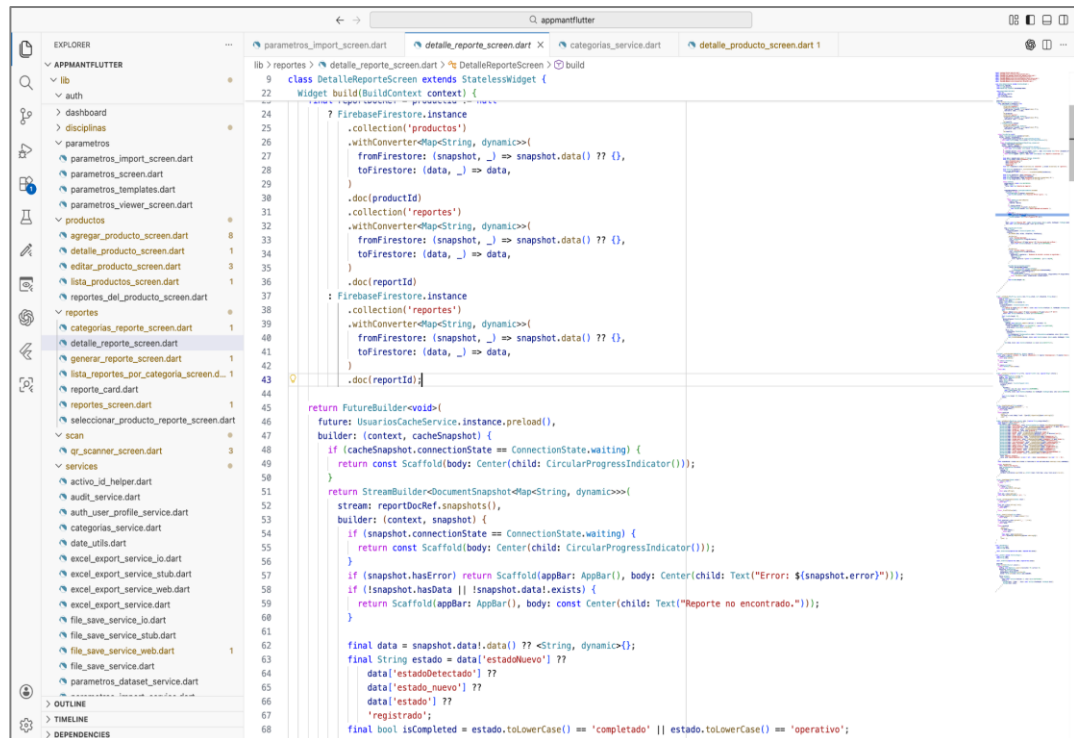
```

15 class SeleccionProductoReporteScreenState extends State<SeleccionProductoReporteScreen> {
16   Widget build(BuildContext context) {
17     return Scaffold(
18       appBar: AppBar(
19         title: const Text("Seleccion Producto"),
20         titleTextStyle: Theme.of(context).appBarTheme.titleTextStyle,
21       ), // AppBar
22       body: Column(
23         children: [
24           Padding(
25             padding: const EdgeInsets.all(12.0),
26             child: TextField(
27               controller: _searchController,
28               decoration: const InputDecoration(
29                 labelText: 'Buscar por nombre, categoría o QR',
30                 prefixIcon: Icon(Icons.search),
31                 border: OutlineInputBorder(),
32               ), // InputDecoration
33               onChanged: (value) => setState(() => _searchQuery = value.toLowerCase().trim()),
34             ), // TextField
35           ), // Padding
36           Expanded(
37             child: StreamBuilder<QuerySnapshot<Map<String, dynamic>>>(
38               stream: FirebaseFirestore.instance
39                 .collection('productos')
40                 .withConverter<Map<String, dynamic>>{
41                   fromFirestore: (snapshot, _) => snapshot.data() ?? {},
42                   toFirestore: (data, _) => data,
43                 }
44                 .where('categoria', isEqualTo: widget.categoriaFilter)
45                 .snapshots(),
46             builder: (context, snapshot) {
47               if (snapshot.connectionState == ConnectionState.waiting) {
48                 return const Center(child: CircularProgressIndicator());
49               }
50               final docs = snapshot.data?.docs ?? [];
51               if (docs.isEmpty) {
52                 return const Center(child: Text("No hay productos en esta categoría."));
53               }
54               final filteredDocs = docs.where((doc) {
55                 final data = doc.data();
56                 final nombre = (data['nombre'] ?? '').toLowerCase();
57                 final categoria = (data['categoria'] ?? '').toLowerCase();
58                 final codigoQR = doc.id.toLowerCase();
59               });
60             },
61           ),
62         ],
63       ),
64     );
65   }
66 }

```

Nota. visual estudio code (2026).

La figura muestra la implementación del archivo seleccionar_producto_reporte_screen.dart dentro del módulo reportes/ del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este archivo permitió seleccionar el activo al cual se asociará el reporte de mantenimiento, realizando una consulta a la colección productos en Cloud Firestore para mostrar el listado de activos disponibles. A través de esta interfaz, el usuario pudo buscar y seleccionar el activo correspondiente, garantizando que el reporte generado quede correctamente vinculado al elemento específico del sistema. Esta implementación permitió establecer de manera adecuada la relación un activo – múltiples reportes, evitando errores de asociación y facilitando el seguimiento histórico de las incidencias registradas para cada activo.

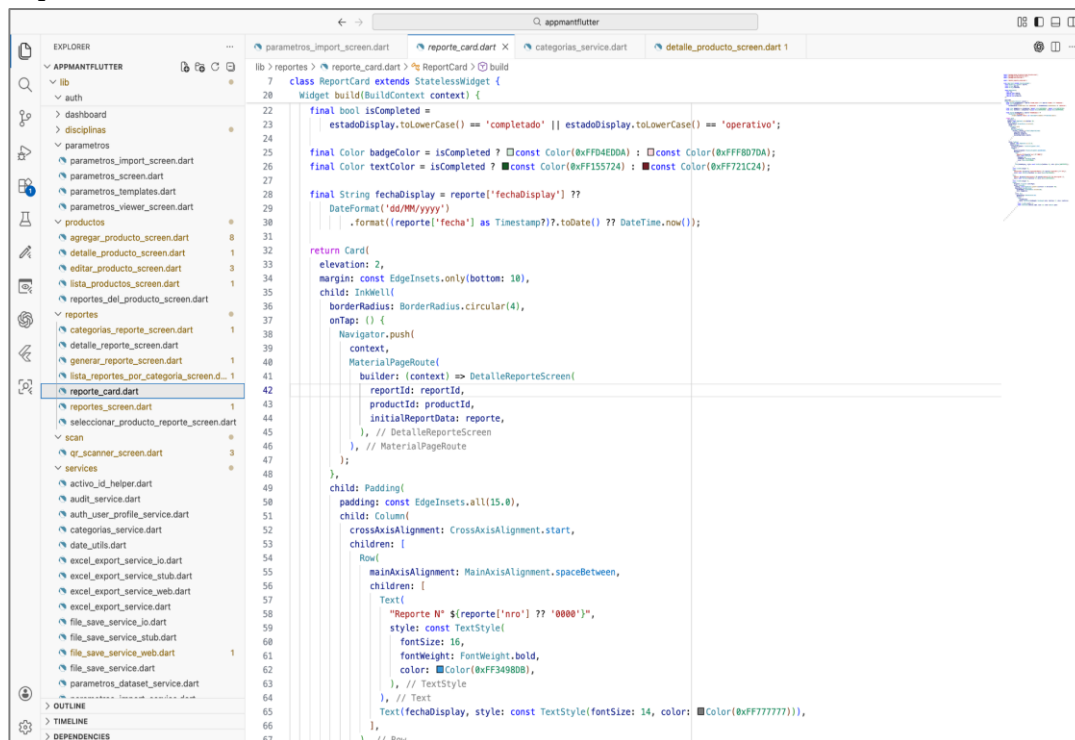
Figura 28*Detalle_reporte_screen.dart*

Nota. visual estudio code (2026).

La figura muestra la implementación del archivo detalle_reporte_screen.dart dentro del módulo reportes/ del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este archivo permitió visualizar la información completa de un reporte de mantenimiento específico, consultando los datos almacenados en Cloud Firestore. A través de esta interfaz se mostraron los datos descriptivos del reporte, incluyendo el estado previo y posterior del activo, la evidencia fotográfica registrada, la fecha de intervención y el usuario responsable del registro. Esta funcionalidad permitió realizar un seguimiento detallado de las incidencias registradas, garantizando la trazabilidad histórica de las intervenciones de mantenimiento dentro del sistema.

Figura 29

Reporte_card.dart



```

lib > reports > reporte_card.dart > %ReportCard% build
7 class ReportCard extends StatelessWidget {
28 Widget build(BuildContext context) {
22   final bool isCompleted =
23     estadoDisplay.toLowerCase() == 'completado' || estadoDisplay.toLowerCase() == 'operativo';
24   final Color badgeColor = isCompleted ? const Color(0xFF04E0DA) : const Color(0xFFFF8070DA);
25   final Color textColor = isCompleted ? const Color(0xFF155724) : const Color(0xFF721C24);
26   final String fechaDisplay = reporte['fechaDisplay'] ??
27     DateFormat('dd/MM/yyyy')
28     .format((reporte['fecha'] as Timestamp?)?.toDate() ?? DateTime.now());
29
30   return Card(
31     elevation: 2,
32     margin: const EdgeInsets.only(bottom: 10),
33     child: InkWell(
34       borderRadius: BorderRadius.circular(4),
35       onTap: () {
36         Navigator.push(
37           context,
38           MaterialPageRoute(
39             builder: (context) => DetalleReporteScreen(
40               reportId: reportId,
41               productId: productId,
42               initialReportData: reporte,
43             ), // DetalleReporteScreen
44           ), // MaterialPageRoute
45         );
46       },
47     ),
48     child: Padding(
49       padding: const EdgeInsets.all(15.0),
50       child: Column(
51         crossAxisAlignment: CrossAxisAlignment.start,
52         children: [
53           Row(
54             mainAxisAlignment: MainAxisAlignment.spaceBetween,
55             children: [
56               Text(
57                 "Reporte N° ${reporte['nro'] ?? '0000'}",
58                 style: const TextStyle(
59                   fontSize: 16,
60                   fontWeight: FontWeight.bold,
61                   color: Color(0xFF3498DB),
62                 ), // TextStyle
63               ), // Text
64               Text(fechaDisplay, style: const TextStyle(fontSize: 14, color: Color(0xFF7F7F7F))),
65             ], // Row
66           ),
67         ],
68       ),
69     ),
70   );
71 }

```

Nota. visual estudio code (2026).

La figura muestra la implementación del archivo `reporte_card.dart` dentro del módulo `reportes/` del aplicativo `App-Mantenimiento`, visualizada en el entorno de desarrollo `Visual Studio Code`. Este archivo implementó un widget reusable encargado de representar visualmente la información de un reporte dentro de las listas del sistema. A través de este componente se encapsuló la estructura visual de cada reporte, mostrando datos relevantes como el identificador del reporte, la fecha de registro y el estado asociado. Su implementación permitió mantener un diseño uniforme en las diferentes vistas donde se muestran reportes, además de facilitar la reutilización del código en múltiples pantallas del aplicativo. Este enfoque demostró la aplicación de buenas prácticas de desarrollo, al promover la modularidad del sistema, la reutilización de componentes y la separación entre la lógica del sistema y la interfaz de usuario.

Integración con Base de Datos: El módulo `reportes` interactuó principalmente con las colecciones `reportes`, `productos` y `categorías` dentro de la base de datos `Cloud Firestore`. A través de estas colecciones se gestionó la información relacionada con las incidencias de mantenimiento registradas en el sistema. En este módulo se implementaron operaciones de

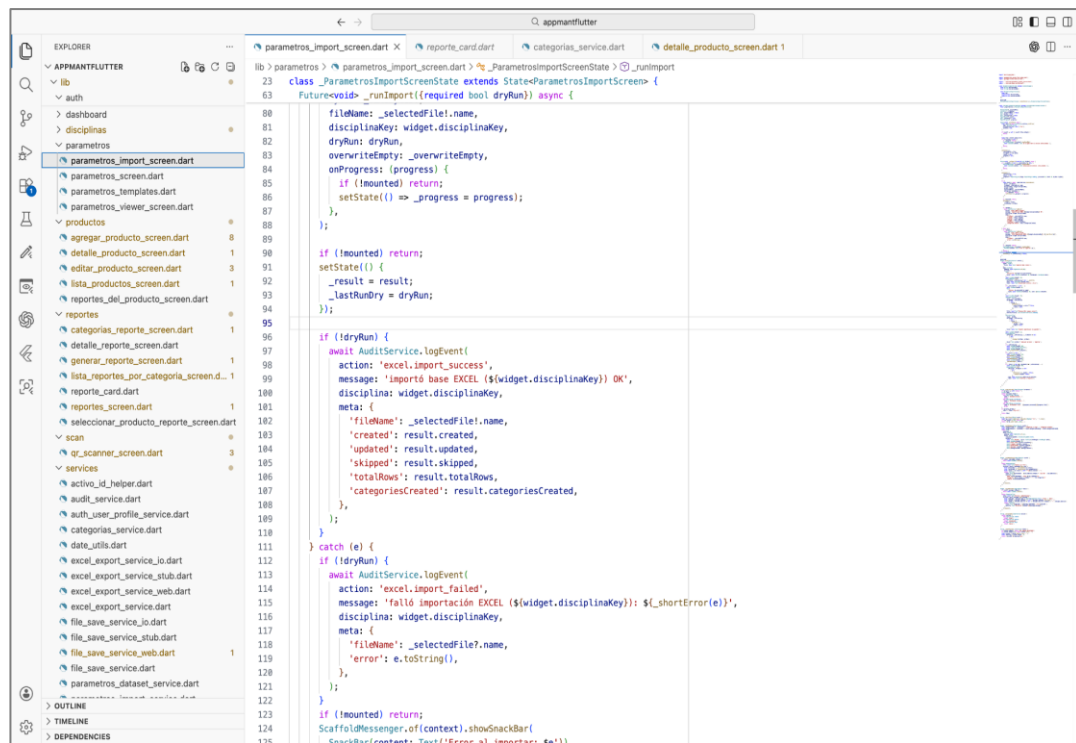
creación de reportes (Create), consulta de reportes registrados (Read), actualización del estado de los activos (Update) y consultas filtradas según disciplina y categoría. Estas operaciones permitieron administrar de manera estructurada la información de mantenimiento y mantener sincronizados los reportes con el estado operativo de los activos.

Impacto en la Arquitectura General: Este módulo convirtió el modelo digital en un sistema dinámico, ya que permitió registrar incidencias reales detectadas en campo, actualizar el estado operativo de los activos y generar un historial de intervenciones de mantenimiento. Asimismo, la información registrada alimentó el Dashboard del sistema, facilitando la visualización y análisis de las condiciones de los activos. De igual manera, los datos generados pudieron ser retroalimentados hacia el entorno BIM en Revit, permitiendo mantener actualizado el modelo digital. De esta manera, este módulo representó el punto en el que la digitalización se transformó en una gestión activa del mantenimiento

Módulo de parámetros (parámetros/)

Figura 30

Estructura interna del módulo Parámetros



Nota. visual estudio code (2026).

La figura muestra la estructura interna del módulo parametros/ del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. En este módulo se implementaron archivos como `parametros_import_screen.dart`, `parametros_screen.dart`, `parametros_templates.dart` y `parametros_viewer_screen.dart`, los cuales permitieron gestionar la importación, visualización y administración de los parámetros provenientes del modelo BIM. A través de estos componentes, el sistema permitió importar los parámetros exportados desde Revit mediante Dynamo, organizarlos en plantillas y visualizar la información técnica asociada a los activos dentro del aplicativo. Esta estructura evidenció que el sistema no solo gestiona los activos, sino que también administró dinámicamente los parámetros del modelo BIM, facilitando la interoperabilidad entre el entorno de modelado y la aplicación de gestión de mantenimiento.

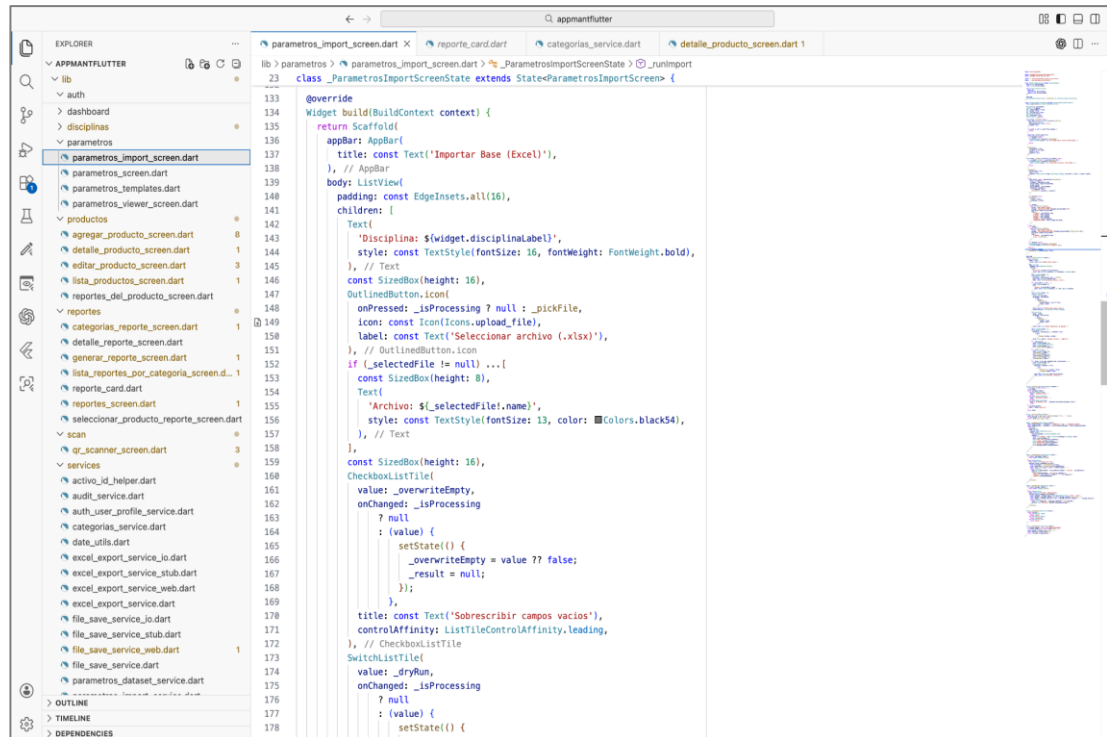
Rol del Módulo Parámetros en la Arquitectura General: El módulo parámetros implementó el proceso de importación y gestión de los parámetros provenientes del modelo BIM. En primer lugar, permitió importar los parámetros desde un archivo Excel exportado desde Revit mediante Dynamo, posteriormente se realizó la validación de la estructura de los datos para asegurar su correcta organización. Asimismo, el sistema ejecutó una simulación de importación (dry run) con el fin de verificar posibles errores antes de realizar la carga definitiva. Una vez validada la información, se procedió a la confirmación de la importación, permitiendo además la visualización de los parámetros cargados y la gestión de plantillas de estructura de datos. Gracias a esta implementación, la aplicación no dependió de parámetros fijos codificados, sino que pudo adaptarse dinámicamente a diferentes disciplinas y estructuras de modelo. Desde el punto de vista de la metodología BIM, este módulo constituyó el núcleo de interoperabilidad entre el modelo digital desarrollado en Revit y el sistema de gestión implementado en el aplicativo.

Flujo de Interoperabilidad Implementado: El módulo parámetros soportó un flujo completo de interoperabilidad entre el entorno BIM y el aplicativo, el cual siguió la secuencia: Revit → Dynamo → Excel → Aplicación → Firebase → Aplicación → Dynamo → Revit. En este proceso, los parámetros de los elementos modelados en Revit fueron exportados mediante Dynamo a un archivo Excel, el cual contenía información técnica de cada elemento, como identificador, disciplina, ubicación, nivel, estado y otros atributos del modelo. Posteriormente, el módulo parámetros/ del aplicativo interpretó la estructura de este archivo, procesando la información para convertir los datos en documentos almacenados dentro de la

base de datos Firebase. De esta manera, el sistema permitió integrar la información del modelo BIM con la aplicación de gestión de mantenimiento, facilitando el intercambio y actualización de datos entre ambas plataformas. En el análisis de Archivos Internos: Se tuvo los siguientes:

Figura 31

Parametros_import_screen.dart



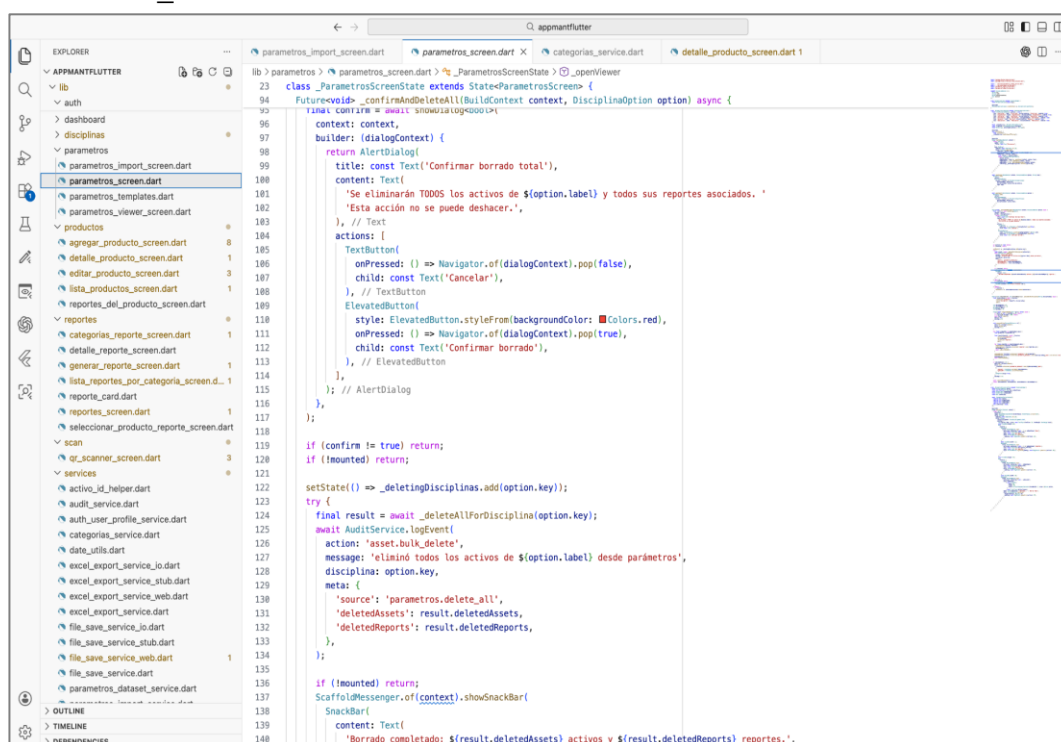
Nota. visual estudio code (2026).

La figura muestra la implementación del archivo *parametros_import_screen.dart* dentro del módulo *parametros/* del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este archivo constituyó el componente principal para la importación de parámetros provenientes de archivos Excel, previamente exportados desde el modelo BIM mediante Dynamo. En el código se observaron elementos como *_isProcessing*, *LinearProgressIndicator*, *_progressLabel* y el método *_runImport(dryRun: false)*, los cuales permitieron gestionar el proceso de importación y mostrar el progreso de la operación. Asimismo, se implementaron funciones como *_buildSummary(_result)*, *_buildRowChanges(_result)* y *_buildMessages(_result)*, que permitieron visualizar un resumen de los datos, registrar los cambios detectados y mostrar mensajes del proceso antes de confirmar la carga definitiva. Esta implementación evidenció que el sistema incorporó un

mecanismo de validación previa mediante un modo de simulación (dry run), permitiendo analizar la información antes de realizar la importación final. De esta manera, el usuario pudo verificar los datos, revisar los cambios detectados y confirmar manualmente la operación, garantizando mayor control y seguridad durante la integración de parámetros del modelo BIM con la base de datos del aplicativo.

Figura 32

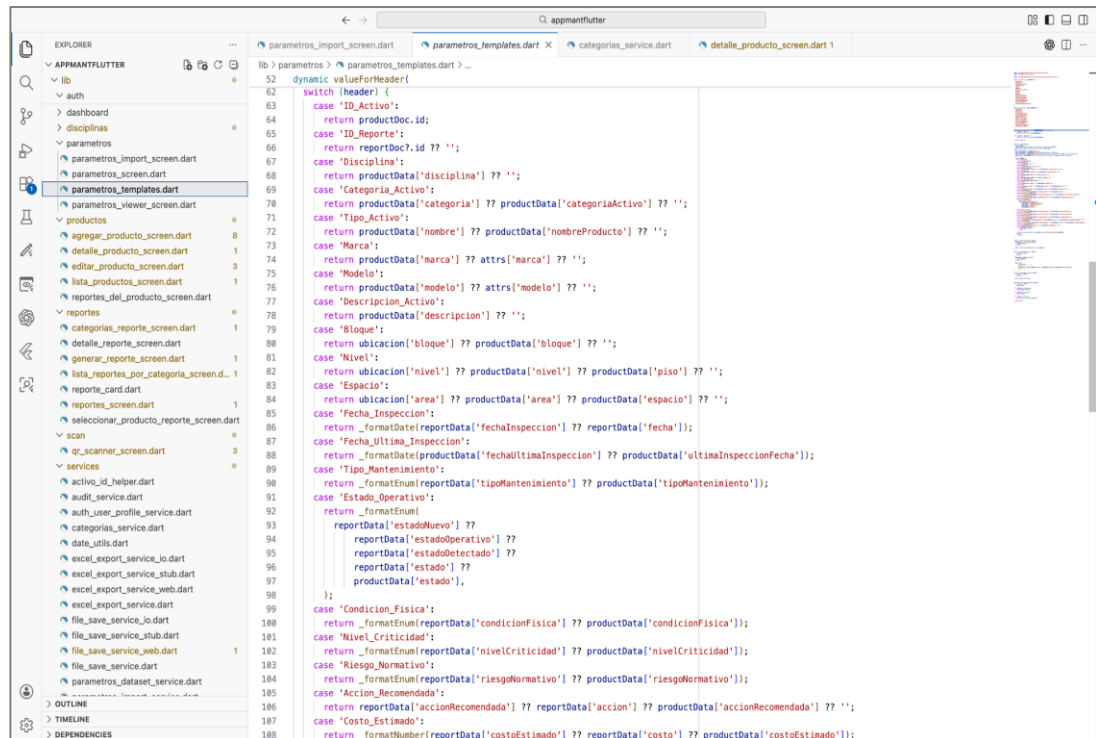
Parametros_screen.dart



Nota. visual estudio code (2026).

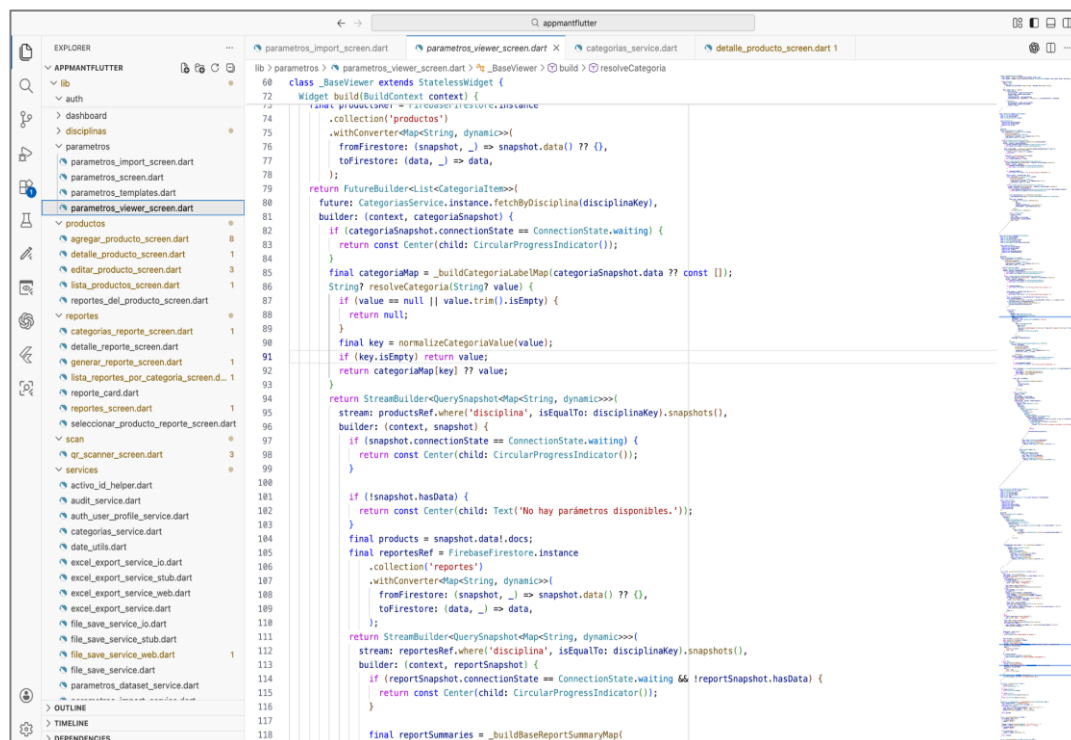
La figura muestra la implementación del archivo *parametros_screen.dart* dentro del módulo *parámetros/* del aplicativo *App-Mantenimiento*, visualizada en el entorno de desarrollo *Visual Studio Code*. Este archivo actuó como la pantalla principal del módulo de gestión de parámetros, funcionando como el punto de acceso administrativo para la administración de la información proveniente del modelo BIM. A través de esta interfaz, el sistema permitió navegar entre las diferentes disciplinas técnicas, visualizar los conjuntos de datos (datasets) cargados, acceder al módulo de importación de parámetros y gestionar configuraciones relacionadas con la estructura de datos del modelo. De esta manera, esta pantalla centralizó las operaciones de administración de parámetros, facilitando la gestión y organización de la información importada desde el entorno BIM.

Figura 33

Parametros_templates.dart

Nota. visual estudio code (2026).

La figura muestra la implementación del archivo `parametros_templates.dart` dentro del módulo `parámetros/` del aplicativo `App-Mantenimiento`, visualizada en el entorno de desarrollo Visual Studio Code. Este archivo definió las plantillas de estructura y los esquemas de columnas esperados en los archivos Excel utilizados para la importación de parámetros provenientes del modelo BIM. En este componente se establecieron las listas de encabezados (headers) y la configuración de las columnas requeridas, permitiendo que el sistema verifique que los archivos importados contengan las columnas, claves y tipos de datos correctos antes de ser procesados. De esta manera, el aplicativo pudo validar la integridad y consistencia de la información importada. Desde el punto de vista de ingeniería de software, este archivo funcionó como una capa de normalización y control de estructura de datos, asegurando que la información proveniente del modelo BIM mantuviera un formato adecuado para su almacenamiento y gestión dentro del sistema.

Figura 34*Parámetros_viewer_screen.dart*

Nota. visual estudio code (2026).

La figura muestra la implementación del archivo *parámetros_viewer_screen.dart* dentro del módulo *parámetros/* del aplicativo *App-Mantenimiento*, visualizada en el entorno de desarrollo Visual Studio Code. Este archivo permitió visualizar los parámetros previamente importados desde el modelo BIM y almacenados en la base de datos Cloud Firestore. A través de esta interfaz el sistema mostró la información de los parámetros en una estructura organizada, permitiendo revisar los datos importados, verificar la correcta carga de los registros y consultar la información técnica asociada a cada activo. Esta funcionalidad facilitó la validación y control de los parámetros integrados al sistema, permitiendo realizar una revisión técnica de la información proveniente del modelo BIM. De esta manera, este componente constituyó una herramienta clave para la auditoría y verificación de los datos importados, garantizando la integridad de la información utilizada en el sistema de gestión de mantenimiento.

Importancia Técnica del Módulo: El módulo *parámetros* cumplió un papel fundamental dentro de la arquitectura del sistema, ya que permitió que la aplicación no dependiera de parámetros definidos de manera rígida en el código, sino que pudiera

adaptarse dinámicamente a diferentes modelos BIM. Gracias a esta implementación, el sistema facilitó la incorporación de nuevas disciplinas y estructuras de datos, permitiendo una mayor escalabilidad en la gestión de la información. Asimismo, este módulo garantizó una sincronización controlada entre los parámetros del modelo BIM y la base de datos del aplicativo. Sin la implementación de este componente, el sistema habría sido rígido y limitado, mientras que con su integración se logró un sistema flexible, dinámico y adaptable a diferentes configuraciones de modelos BIM.

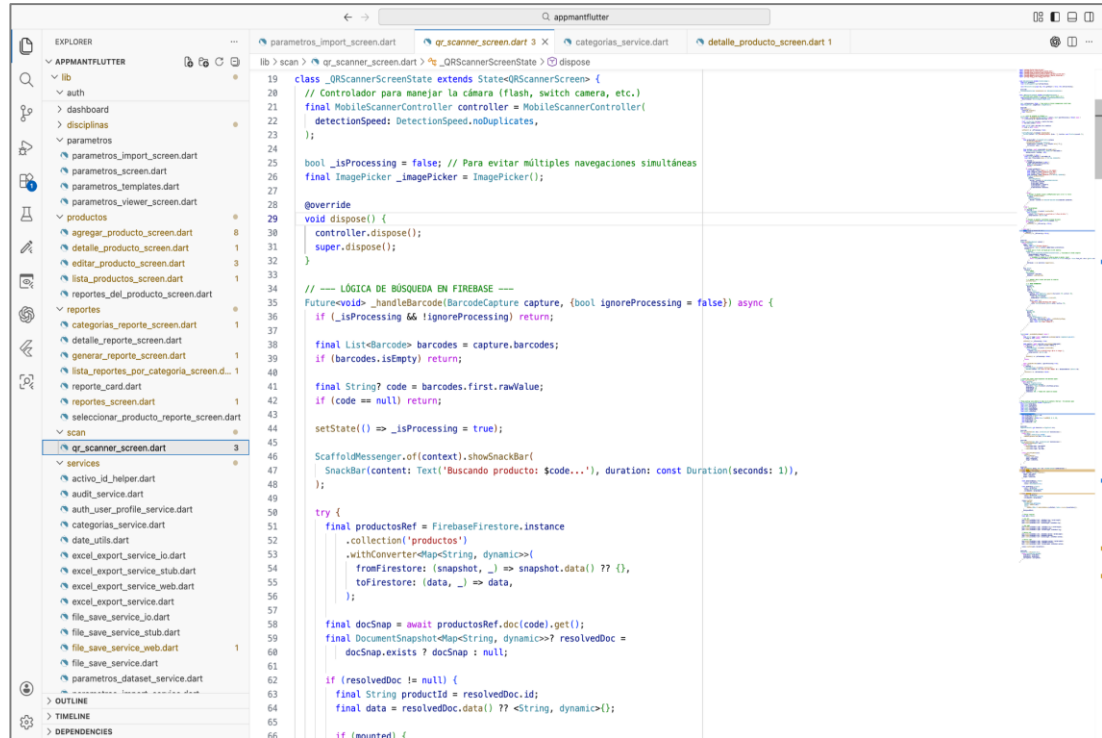
Control de Calidad de Datos: El sistema incorporó mecanismos de control de calidad de datos mediante la implementación de validaciones previas, simulación del proceso de importación, generación de un resumen de cambios y confirmación manual antes de la carga definitiva de la información. Estas funcionalidades permitieron verificar la consistencia y estructura de los datos antes de su almacenamiento en la base de datos, evitando errores durante el proceso de integración. De esta manera, el sistema fue diseñado considerando la integridad y confiabilidad de la información, lo cual resulta fundamental dentro de una implementación técnica orientada a la gestión de datos provenientes de un entorno BIM.

Relación con el Modelo BIM: El módulo parámetros constituyó el punto de entrada y salida de información entre el modelo BIM desarrollado en Revit y el sistema de gestión de mantenimiento implementado en el aplicativo. A través de este componente fue posible actualizar el estado de los activos, sincronizar los parámetros modificados y mantener la coherencia entre la información del modelo digital y las condiciones operativas registradas en campo. De esta manera, el módulo permitió asegurar una integración continua entre el entorno BIM y la aplicación de gestión, facilitando la actualización y consistencia de los datos en ambas plataformas.

Módulo de escaneo (scan/)

Figura 35

Implementación del módulo de escaneo QR.



Nota. visual estudio code (2026).

La figura muestra la implementación del archivo `qr_scanner_screen.dart` dentro del módulo `scan/` del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este archivo permitió integrar la funcionalidad de escaneo de códigos QR para identificar rápidamente los activos registrados en el sistema durante las actividades de mantenimiento en campo. A través de este componente, el sistema capturó el código QR asociado a un activo, realizó la consulta correspondiente en la base de datos Cloud Firestore y, en caso de encontrar coincidencia, redirigió automáticamente a la pantalla de detalle del activo. En caso contrario, el sistema mostró un mensaje indicando que el producto no se encontraba registrado en la base de datos. Esta funcionalidad facilitó la identificación inmediata de los activos, reduciendo el tiempo de búsqueda y permitiendo acceder rápidamente a la información técnica y a los reportes asociados. De esta manera, el módulo de escaneo incorporó una herramienta clave para optimizar las actividades de mantenimiento realizadas en campo.

Rol del Módulo Scan: Permitió la identificación rápida de los activos mediante la lectura de códigos QR asociados a cada elemento. A través de esta funcionalidad, el sistema capturó el código del activo y redirigió automáticamente a la pantalla de detalle correspondiente, donde se mostró la información técnica y los reportes asociados. De esta manera, el módulo facilitó el acceso inmediato a los datos del activo, eliminando la necesidad de realizar búsquedas manuales dentro del sistema y optimizando las actividades de mantenimiento realizadas en campo.

Análisis Técnico del Código: Se tiene lo siguiente:

Figura 36

Análisis Técnico del Código.

```
89         } else {  
90             // Navegar al detalle (usamos pushReplacement para cerrar la cámara)  
91             Navigator.pushReplacement(  
92                 context,  
93                 MaterialPageRoute(  
94                     builder: (context) => DetalleProductoScreen(productId: productId),  
95                 ), // MaterialPageRoute  
96             );  
97         }  
98     }  
99     } else {
```

Nota. visual estudio code (2026).

La figura muestra un fragmento del código implementado en el módulo de escaneo QR del aplicativo App-Mantenimiento, donde se gestionó el flujo de identificación de activos mediante códigos QR. En este proceso, el sistema escanea el código QR y obtuvo que el identificador del activo (productId). Posteriormente, el aplicativo realizó una consulta en la base de datos Cloud Firestore para verificar si el activo se encuentra registrado en el sistema. Si el activo existe en la base de datos, el sistema redirige automáticamente a la pantalla de detalle del producto mediante la función `Navigator.pushReplacement`, permitiendo visualizar la información del activo. En caso de que el activo no exista, el sistema muestra un mensaje indicando que el producto no se encuentra registrado en la base de datos.

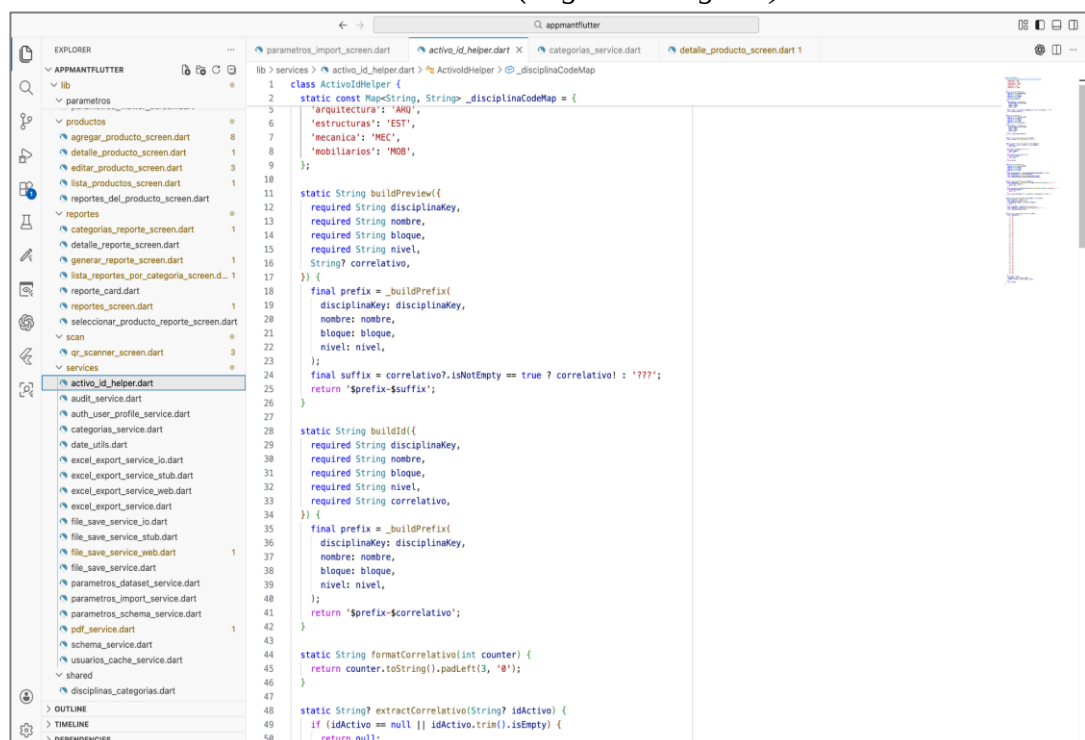
Impacto Operativo: El módulo scan convirtió el aplicativo en una herramienta operativa para el trabajo en campo, ya que permitió la identificación inmediata de los activos mediante la lectura de códigos QR. Esta funcionalidad facilitó el acceso directo a la información del

elemento, incluyendo su historial de mantenimiento, lo que contribuyó a reducir errores humanos durante el proceso de identificación. Asimismo, permitió actualizar de manera rápida el estado de los activos y registrar nuevas incidencias, optimizando las actividades de mantenimiento. Desde el punto de vista de la gestión de mantenimiento, esta funcionalidad resultó altamente eficiente para mejorar la trazabilidad y la rapidez en la toma de decisiones operativas.

Módulo de servicios (services/)

Figura 37

Estructura interna del módulo Services (Lógica de Negocio)



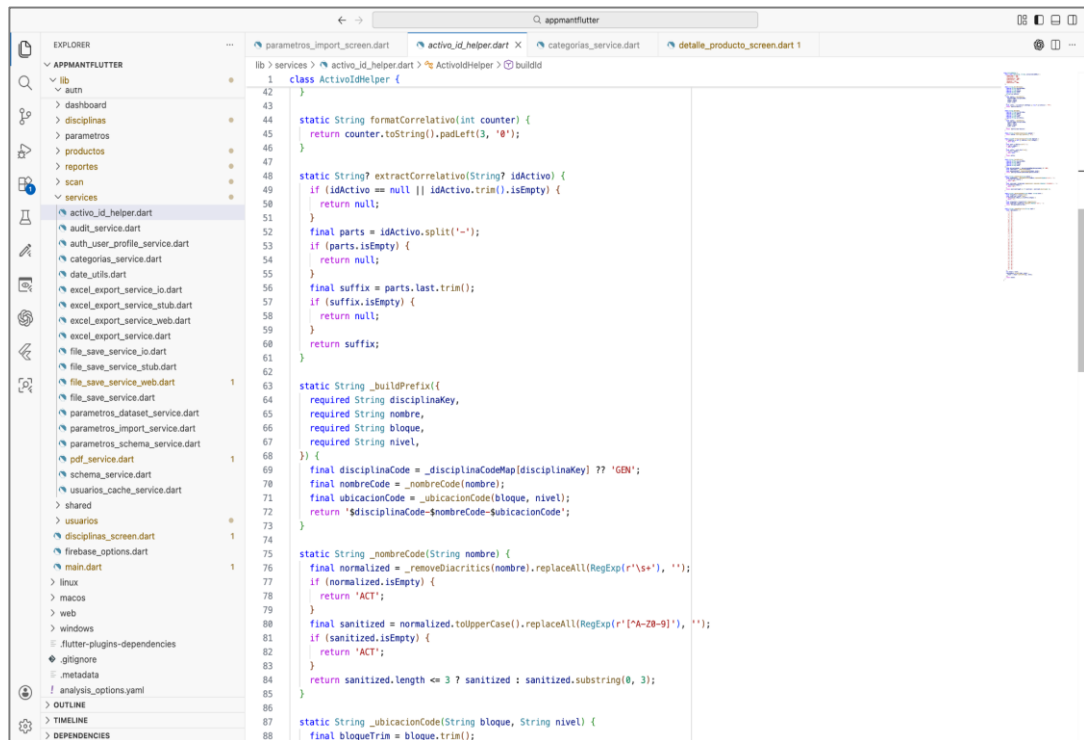
Nota. visual estudio code (2026).

La figura muestra la estructura interna del módulo services/ del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. En este módulo se implementaron diferentes archivos encargados de gestionar la lógica de negocio del sistema, tales como activo_id_helper.dart, audit_service.dart, auth_user_profile_service.dart, categorias_service.dart, date_utils.dart, excel_export_service.dart, file_save_service.dart, parametros_dataset_service.dart, parametros_import_service.dart, parametros_schema_service.dart, pdf_service.dart y

schema_service.dart. Cada uno de estos componentes cumplió funciones específicas relacionadas con la generación de identificadores de activos, auditoría de operaciones, gestión de perfiles de usuario, administración de categorías, manejo de fechas, exportación de datos a Excel, almacenamiento de archivos, procesamiento de parámetros BIM y generación de reportes en PDF. La organización de estos servicios evidenció una separación clara entre las diferentes capas del sistema, donde la interfaz de usuario (screens) se encargó de la interacción con el usuario, la capa services gestionó la lógica del sistema y la capa de persistencia se conectó con las bases de datos y servicios en la nube como Firebase y Supabase. Esta estructura corresponde a un enfoque de arquitectura por capas o arquitectura limpia, lo cual mejoró la mantenibilidad, escalabilidad y organización del aplicativo.

Rol del Módulo Services: Encapsuló toda la lógica de negocio del sistema, cumpliendo un papel fundamental dentro de la arquitectura del aplicativo. Su función principal consistió en separar la lógica de manejo de datos de la interfaz gráfica, permitiendo mantener una organización clara entre las diferentes capas del sistema. A través de este módulo se centralizaron las operaciones de interacción con Firebase, se implementaron las reglas de negocio del aplicativo, y se gestionaron procesos como exportación e importación de datos, generación de identificadores de activos y registro de auditoría de operaciones. Este diseño permitió evitar que la lógica del sistema estuviera mezclada dentro de las pantallas del aplicativo, favoreciendo una arquitectura más ordenada, mantenible y escalable.

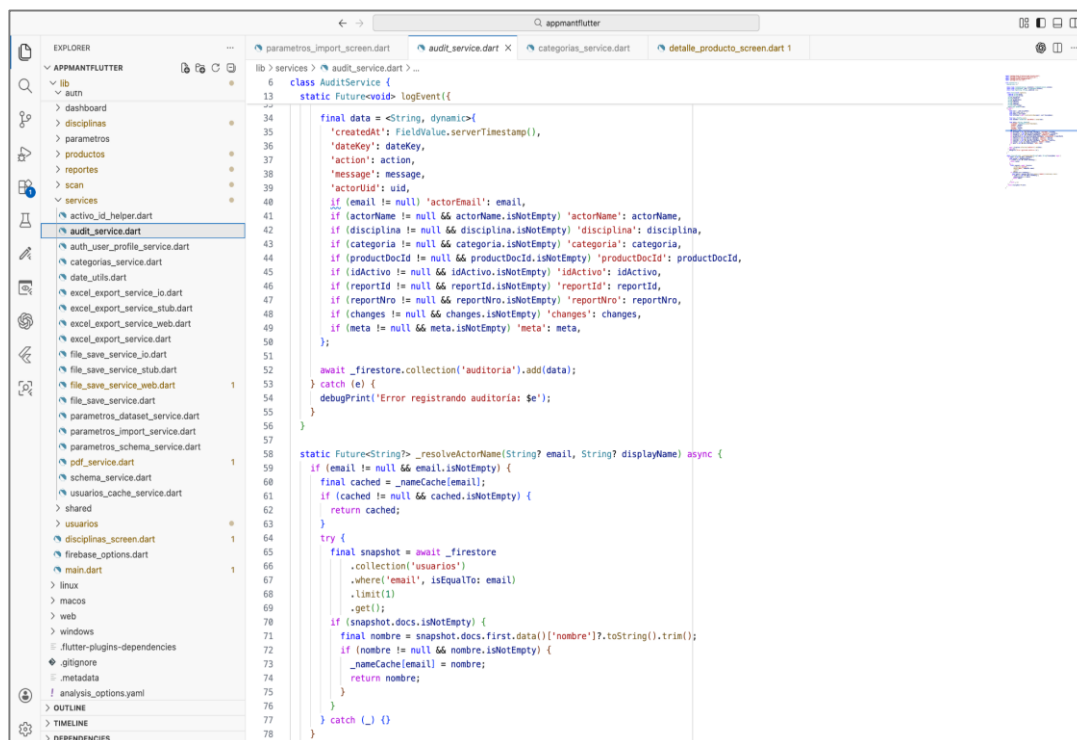
Análisis por Componentes: Tenemos los siguiente:

Figura 38**Activo_id_helper.dart**

```
lib > services > activo_id_helper.dart > ActivoIdHelper > buildId
1 class ActivoIdHelper {
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44 static String formatCorrelativo(int counter) {
45   return counter.toString().padLeft(3, '0');
46 }
47
48 static String? extractCorrelativo(String? idActivo) {
49   if (idActivo == null || idActivo.trim().isEmpty) {
50     return null;
51   }
52   final parts = idActivo.split('-');
53   if (parts.isEmpty) {
54     return null;
55   }
56   final suffix = parts.last.trim();
57   if (suffix.isEmpty) {
58     return null;
59   }
60   return suffix;
61 }
62
63 static String _buildPrefix({
64   required String disciplinaKey,
65   required String nombre,
66   required String bloque,
67   required String nivel,
68 }) {
69   final disciplinaCode = _disciplinaCodeMap[disciplinaKey] ?? 'GEN';
70   final nombreCode = _nombreCode(nombre);
71   final ubicacionCode = _ubicacionCode(bloque, nivel);
72   return '$disciplinaCode-$nombreCode-$ubicacionCode';
73 }
74
75 static String _nombreCode(String nombre) {
76   final normalized = _removeDiacritics(nombre).replaceAll(RegExp(r'\s+'), '');
77   if (normalized.isEmpty) {
78     return 'ACT';
79   }
80   final sanitized = normalized.toUpperCase().replaceAll(RegExp(r'["A-Z0-9]'), '');
81   if (sanitized.isEmpty) {
82     return 'ACT';
83   }
84   return sanitized.length <= 3 ? sanitized : sanitized.substring(0, 3);
85 }
86
87 static String _ubicacionCode(String bloque, String nivel) {
88   final bloqueTrim = bloque.trim();
```

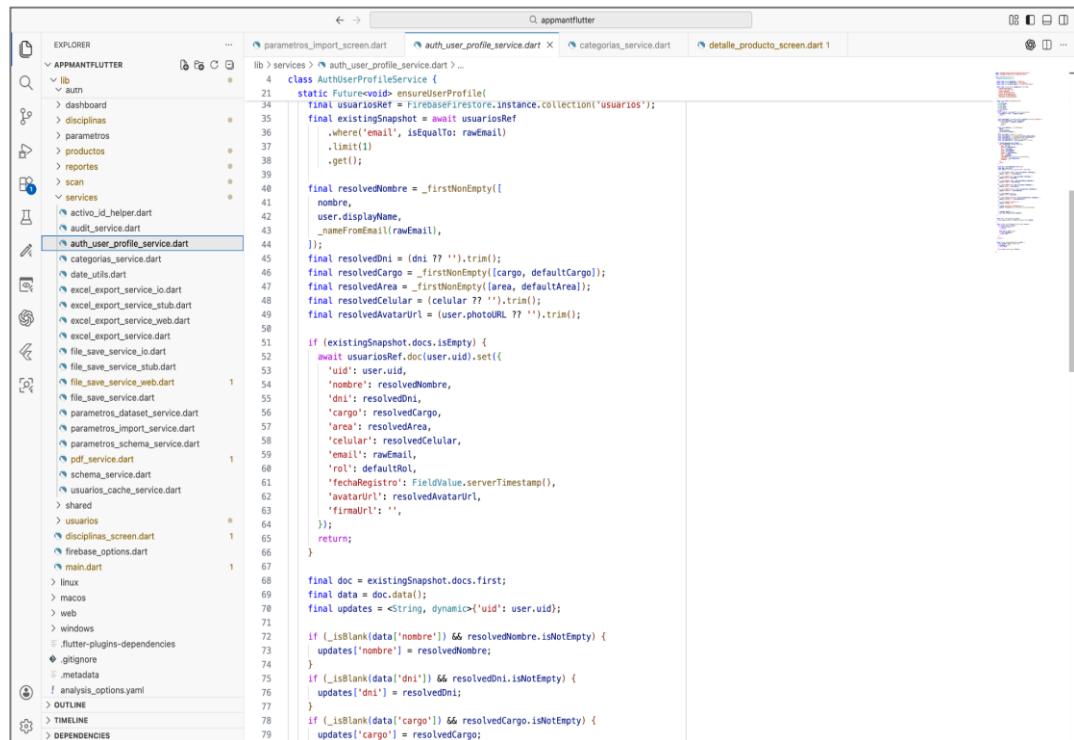
Nota. visual estudio code (2026).

La figura muestra la implementación del archivo activo_id_helper.dart dentro del módulo services/ del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. En este archivo se definió un mapa de códigos de disciplina que permitió asignar abreviaturas específicas a cada especialidad técnica, como eléctricas, sanitarias, arquitectura, estructuras, mecánica y mobiliarios. Este componente se utilizó para generar identificadores estructurados de los activos, construyendo un prefijo basado en información relevante como disciplina, nombre del elemento, bloque, nivel y un correlativo. A través de funciones como buildPreview() y buildId(), el sistema pudo generar identificadores únicos, consistentes y normalizados para cada activo registrado.

Figura 39*Audit_service.dart*

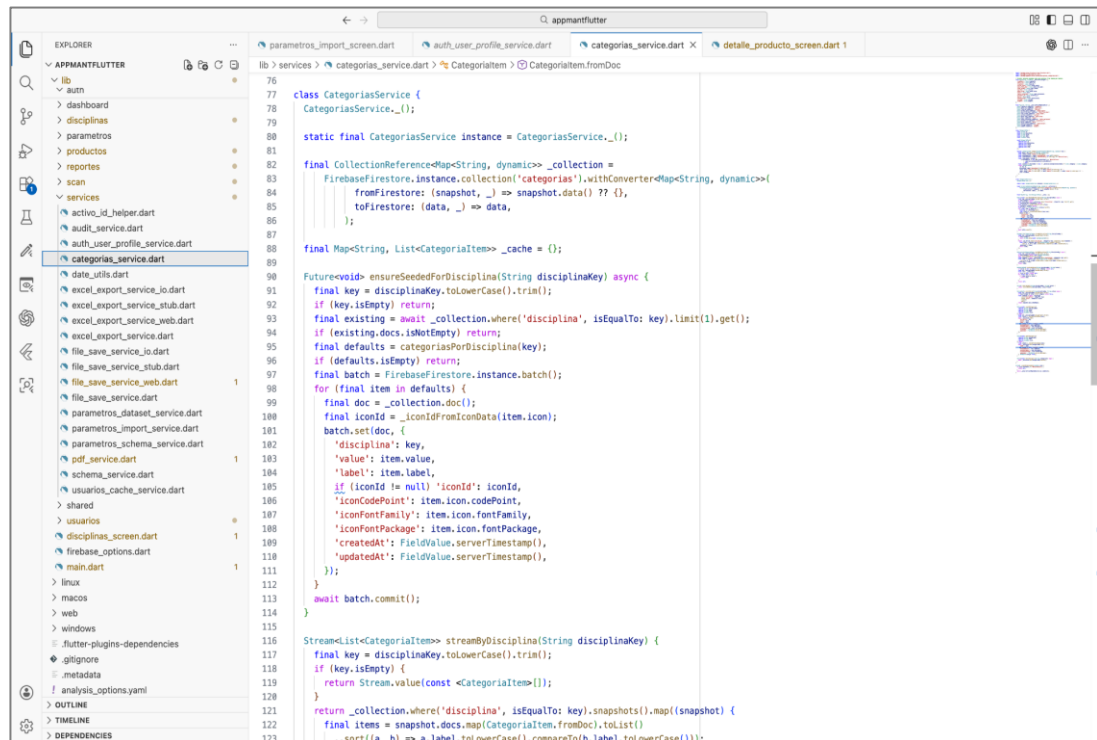
Nota. visual estudio code (2026).

La figura muestra la implementación del archivo `audit_service.dart` dentro del módulo `services/` del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este servicio se encargó de gestionar el registro de eventos y acciones realizadas dentro del sistema, permitiendo almacenar información relevante para el control y seguimiento de las operaciones ejecutadas por los usuarios. A través de este componente se registraron datos como la acción realizada, el usuario responsable, la fecha de ejecución, el módulo involucrado y un mensaje descriptivo del evento. Esta información fue almacenada en la colección `auditoria` dentro de Cloud Firestore, permitiendo mantener un historial de actividades realizadas en el sistema. Desde el punto de vista técnico, esta implementación permitió establecer mecanismos de trazabilidad y control administrativo, facilitando la supervisión de las operaciones del sistema y garantizando mayor transparencia en la gestión de los datos y procesos del aplicativo.

Figura 40*Auth_user_profile_service.dart*

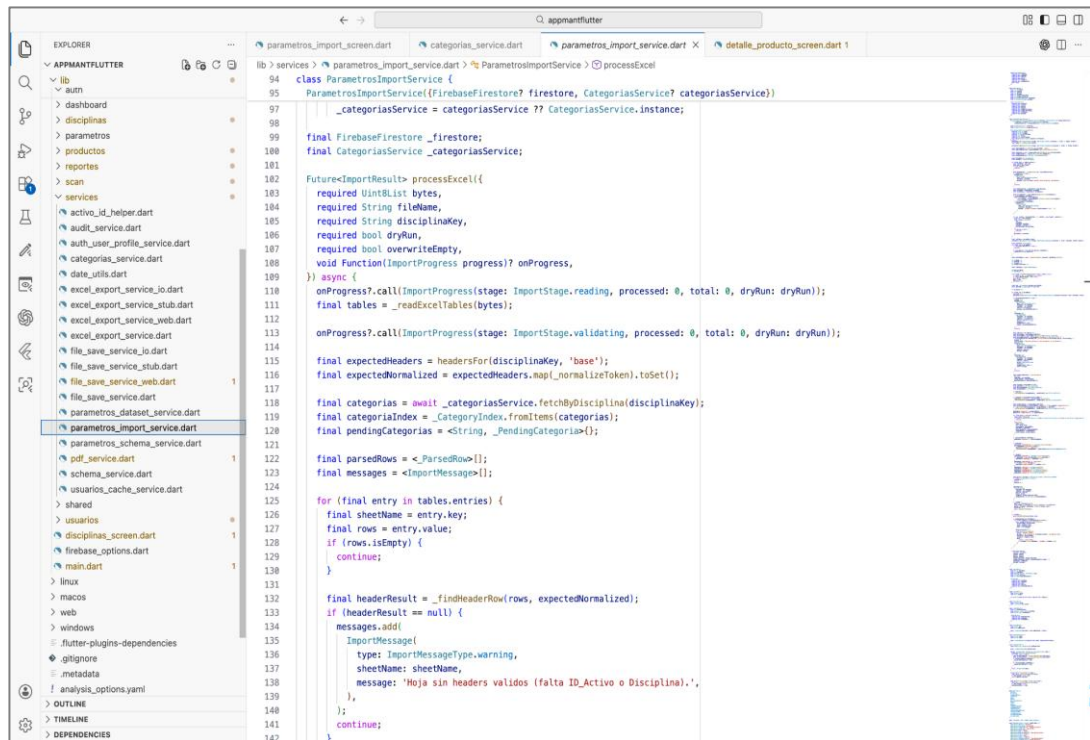
Nota. visual estudio code (2026).

La figura muestra la implementación del archivo `auth_user_profile_service.dart` dentro del módulo `services/` del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este servicio se encargó de gestionar la información del perfil de los usuarios del sistema, permitiendo consultar, registrar y actualizar los datos personales asociados a cada cuenta. A través de este componente se administraron aspectos relacionados con roles, permisos y datos de identificación del usuario, tales como nombre, cargo, área o información de contacto. Esta implementación permitió separar el proceso de autenticación gestionado por Firebase Authentication del manejo del perfil extendido almacenado en la base de datos, lo que facilitó una administración más organizada de la información de los usuarios.

Figura 41*Categorias_service.dart*

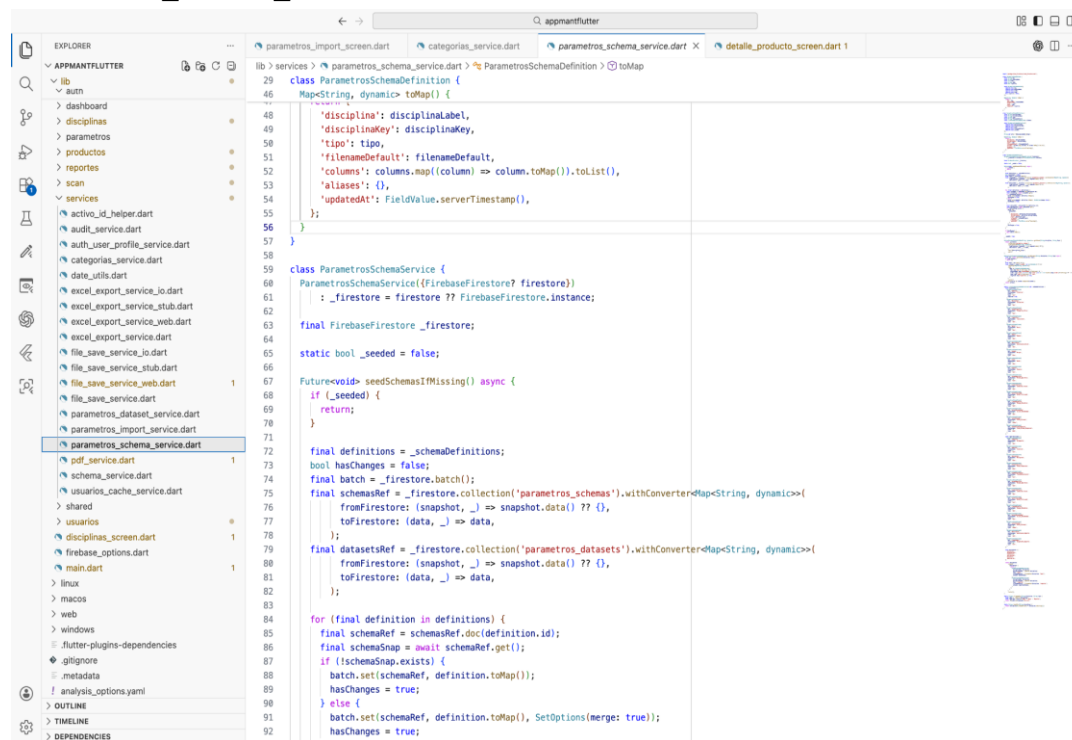
Nota. visual estudio code (2026).

La figura muestra la implementación del archivo `categorias_service.dart` dentro del módulo `services/` del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este servicio se encargó de gestionar las categorías utilizadas para la clasificación de los reportes de mantenimiento, permitiendo organizar las incidencias según la disciplina técnica correspondiente. A través de este componente se implementaron funciones para crear automáticamente las categorías base del sistema, consultar categorías según la disciplina y garantizar la existencia de los registros necesarios antes de ser utilizados. En el código se observa el uso del método `CategoriasService.instance.ensureSeededForDisciplina(...)`, el cual permitió realizar una inicialización automática controlada de las categorías, asegurando que el sistema dispusiera de las estructuras necesarias para el registro de reportes.

Figura 42*Parámetros_import_service.dart*

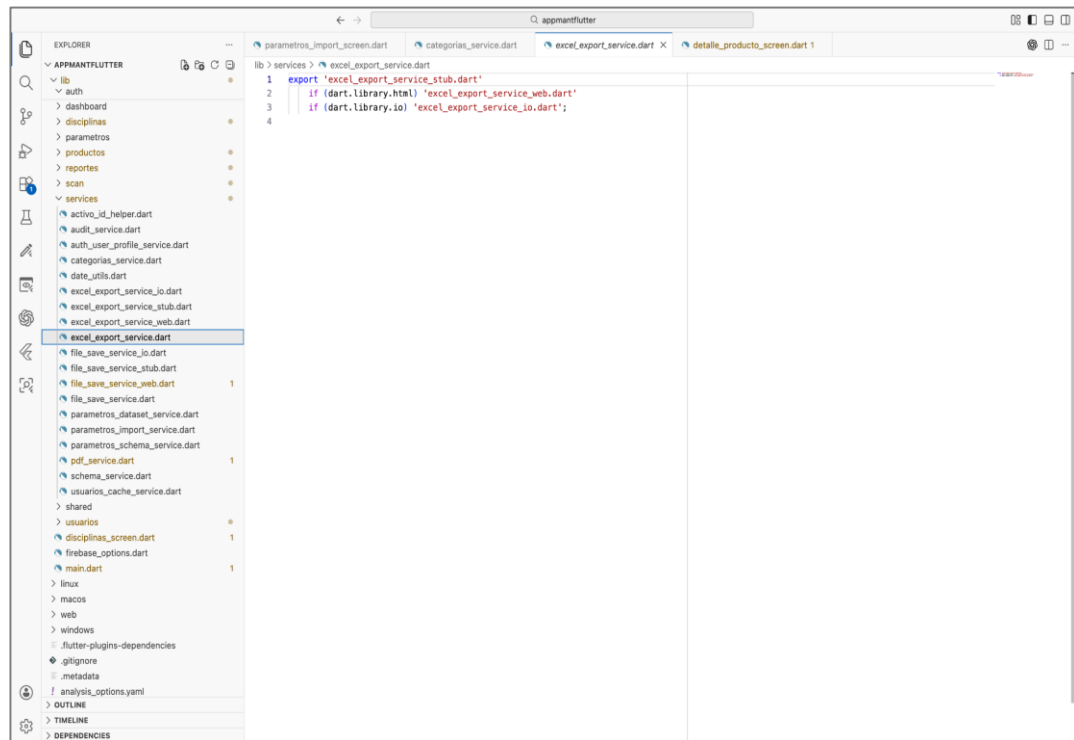
Nota. visual estudio code (2026).

La figura muestra la implementación del archivo `parametros_import_service.dart` dentro del módulo `services/` del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este servicio se encargó de procesar e interpretar los archivos Excel utilizados para la importación de parámetros provenientes del modelo BIM, previamente exportados desde Revit mediante Dynamo. A través de este componente se implementó la lógica necesaria para leer el archivo Excel, validar la estructura de las columnas, interpretar los registros y convertir la información en documentos que posteriormente son almacenados en Cloud Firestore. Durante este proceso el sistema verificó la consistencia de los datos, identificó posibles errores o inconsistencias y registró los cambios detectados antes de realizar la importación definitiva.

Figura 43*Parámetros_schema_service.dart*

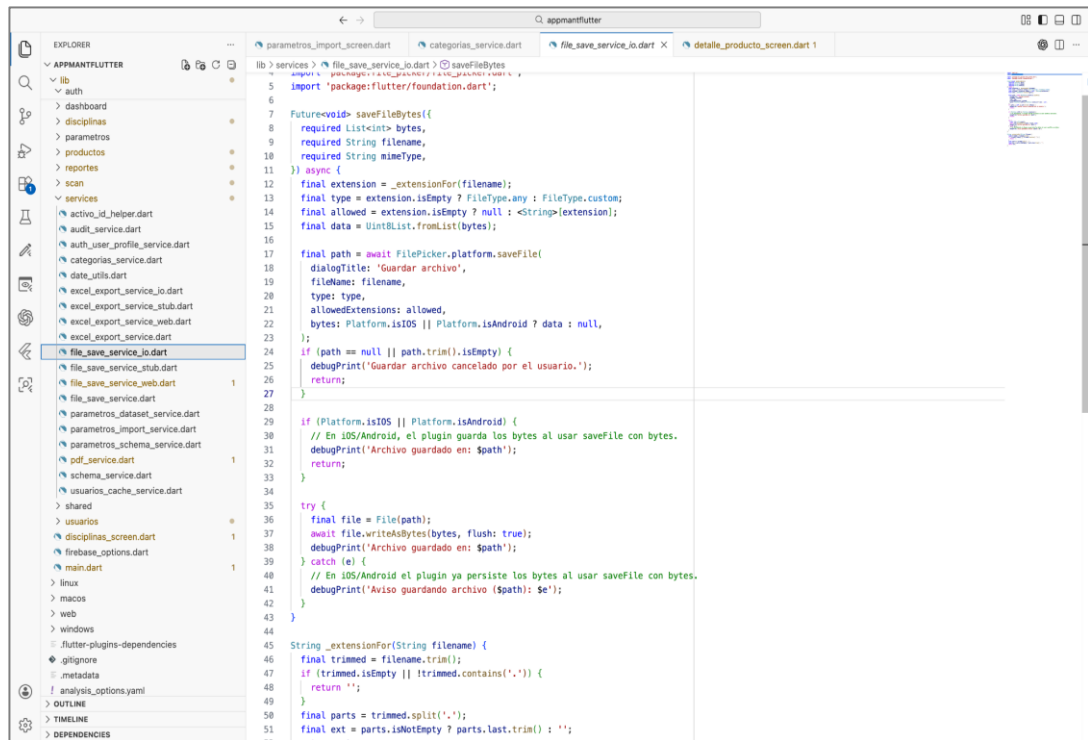
Nota. visual estudio code (2026).

La figura muestra la implementación del archivo `parametros_schema_service.dart` dentro del módulo `services/` del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este servicio se encargó de definir y gestionar los esquemas estructurales de los parámetros utilizados en el sistema, estableciendo la configuración de las columnas y los tipos de datos esperados para cada conjunto de información importada desde el modelo BIM. A través de este componente se implementaron funciones para definir las columnas del esquema, validar los tipos de datos y asegurar la consistencia estructural de la información almacenada en la base de datos. De esta manera, el sistema pudo garantizar que los parámetros importados mantuvieran una estructura organizada y coherente, evitando inconsistencias durante el procesamiento de los datos.

Figura 44*Excel_export_service.dart*

Nota. visual estudio code (2026).

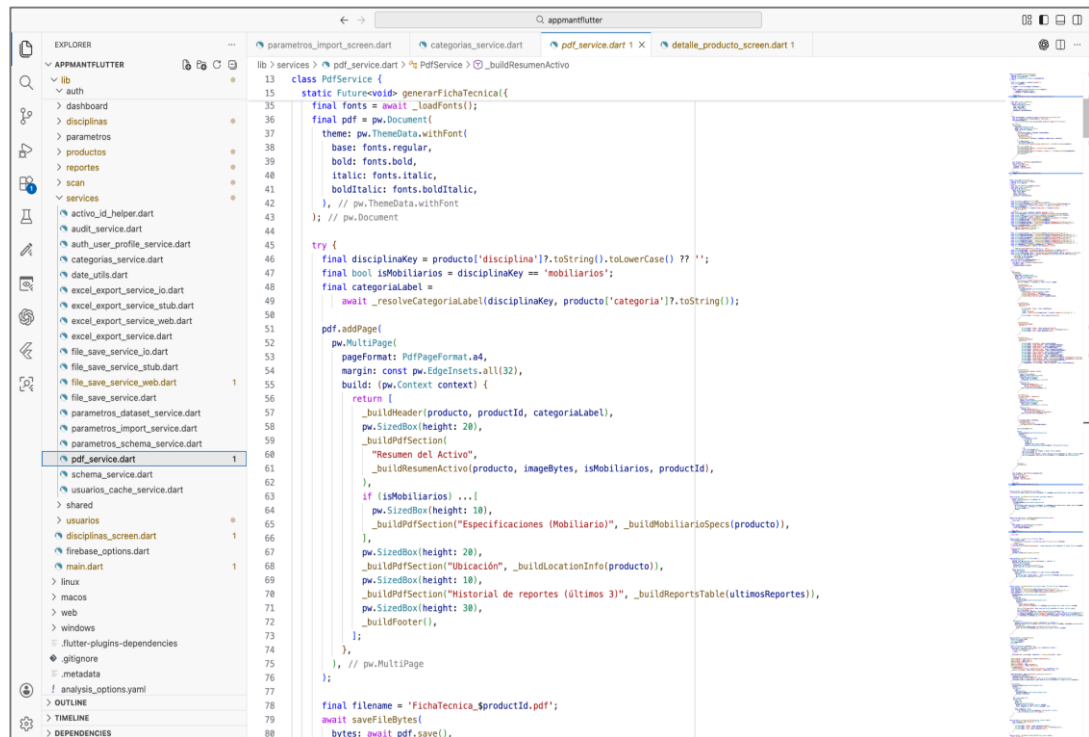
La figura muestra la implementación del archivo `excel_export_service.dart` dentro del módulo `services/` del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este servicio se encargó de exportar la información almacenada en el sistema hacia archivos en formato Excel, permitiendo generar documentos compatibles con los procesos de intercambio de datos utilizados en el entorno BIM. A través de este componente, el sistema generó archivos Excel que pueden ser utilizados por Dynamo para reimportar información actualizada al modelo BIM en Revit. De esta manera, los datos registrados o modificados en el aplicativo, como estados de activos, reportes de mantenimiento y parámetros actualizados, pudieron ser exportados nuevamente para su integración con el modelo digital. Esta funcionalidad permitió cerrar el ciclo de interoperabilidad BIM, ya que la información no solo se importó desde el modelo hacia la aplicación, sino que también pudo ser exportada nuevamente al entorno BIM, manteniendo la coherencia entre el modelo digital y la gestión operativa del mantenimiento.

Figura 45*File_save_service.dart*

Nota. visual estudio code (2026).

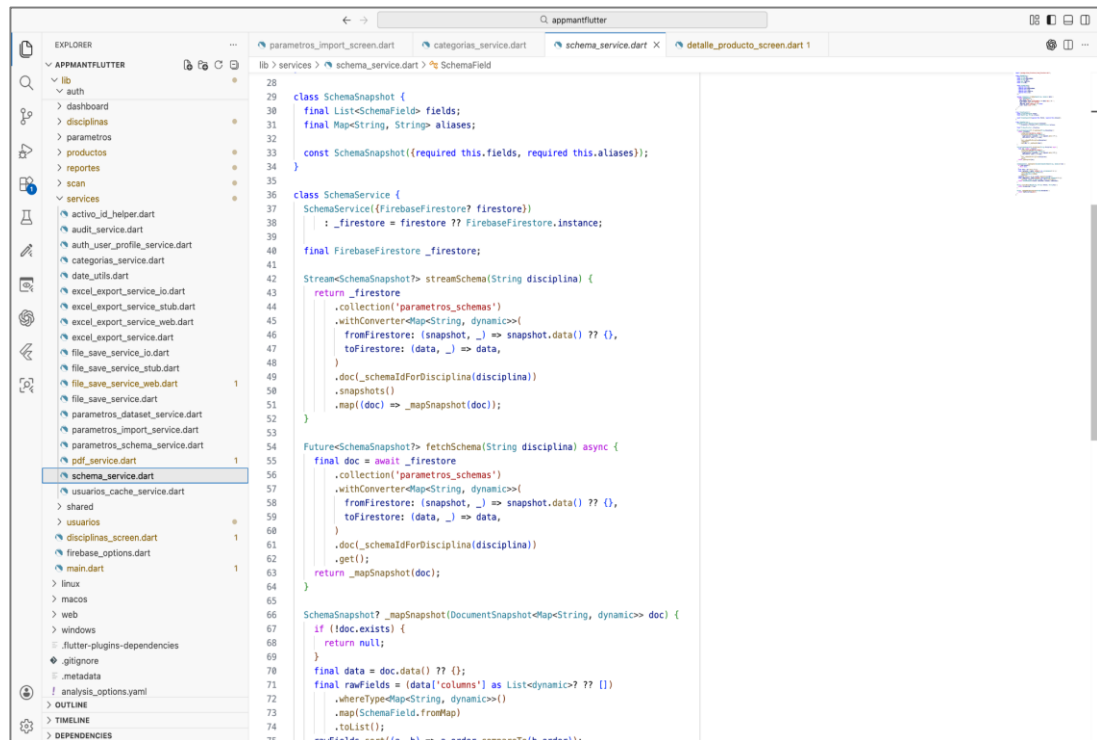
La figura muestra la implementación del archivo `file_save_service.dart` dentro del módulo `services/` del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este servicio se encargó de gestionar el proceso de guardado y descarga de archivos generados por el sistema, permitiendo almacenar documentos como reportes o exportaciones de datos en el dispositivo del usuario. A través de este componente se implementaron funciones para guardar archivos en diferentes plataformas, considerando la compatibilidad entre entornos web y sistemas de entrada/salida (IO) como Android o iOS. De esta manera, el sistema pudo realizar descargas controladas de archivos generados por el aplicativo, asegurando que los datos exportados pudieran ser almacenados correctamente en el dispositivo. Esta implementación evidencia que la aplicación fue diseñada con soporte multiplataforma, permitiendo que las funcionalidades de exportación y almacenamiento de archivos se adapten al entorno en el que se ejecuta el sistema.

Figura 46

Pdf_service.dart

Nota. visual estudio code (2026).

La figura muestra la implementación del archivo pdf_service.dart dentro del módulo services/ del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este servicio se encargó de generar documentos en formato PDF a partir de la información registrada en el sistema, permitiendo crear reportes técnicos asociados a los activos y a las intervenciones de mantenimiento. A través de este componente, el sistema procesó la información del activo, los reportes registrados y las evidencias fotográficas, organizando estos datos en un documento estructurado en formato PDF. De esta manera, fue posible generar documentación técnica exportable, la cual puede ser utilizada como evidencia formal de las actividades de mantenimiento realizadas. Esta funcionalidad añadió un valor profesional al sistema, ya que permitió transformar los registros digitales en informes técnicos formales, facilitando la documentación, presentación y respaldo de la información generada por el aplicativo.

Figura 47*Schema_service.dart*

Nota. visual estudio code (2026).

La figura muestra la implementación del archivo `schema_service.dart` dentro del módulo `services/` del aplicativo App-Mantenimiento, visualizado en el entorno de desarrollo Visual Studio Code. Este servicio se encargó de centralizar la definición de las estructuras de datos y las validaciones generales del sistema, permitiendo gestionar de forma organizada los esquemas utilizados para almacenar y procesar información. En el código se observa la definición de clases como `SchemaField` y `SchemaSnapshot`, las cuales permitieron estructurar los campos que conforman los diferentes conjuntos de datos del sistema. Asimismo, el servicio `SchemaService` interactuó con la base de datos Cloud Firestore para consultar y administrar los esquemas almacenados en la colección correspondiente, asegurando que los datos utilizados por la aplicación mantengan una estructura consistente y validada. De esta manera, este componente formó parte del control estructural del sistema, ya que permitió gestionar los esquemas de datos de manera centralizada, facilitando la validación de la información y garantizando coherencia entre los distintos módulos del aplicativo.

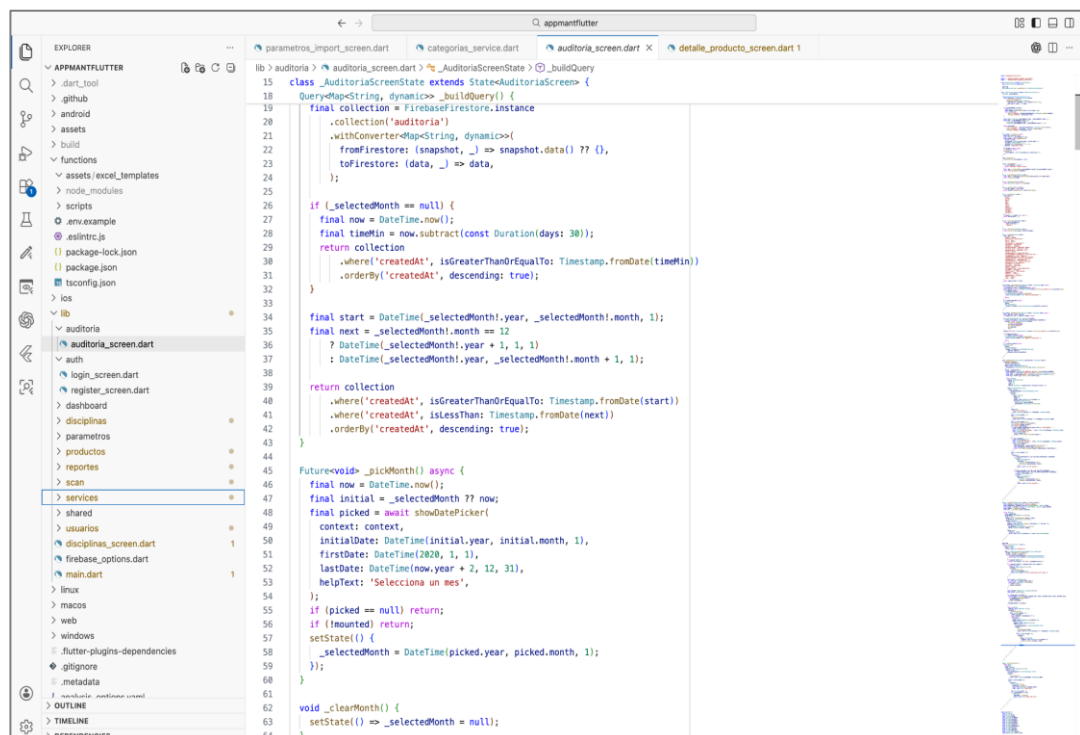
Arquitectura Técnica Implementada: El sistema fue desarrollado siguiendo una arquitectura técnica basada en tres capas, la cual permitió organizar el funcionamiento del

aplicativo de manera estructurada y modular. La primera capa correspondió a la capa de presentación, representada por las pantallas o *screens*, donde se implementó la interfaz de usuario y se gestionan las interacciones con el sistema. La segunda capa estuvo conformada por los servicios, en los cuales se concentró la lógica de negocio, incluyendo el procesamiento de datos, la validación de información y la ejecución de las operaciones principales del sistema. Finalmente, la tercera capa correspondió a la capa de persistencia, encargada de almacenar y gestionar la información mediante servicios en la nube como Firebase y Supabase. La adopción de este modelo arquitectónico permitió mejorar la mantenibilidad del sistema, facilitar su escalabilidad, reducir el acoplamiento entre componentes y posibilitar la implementación de pruebas y futuras ampliaciones, garantizando una estructura de desarrollo más organizada y robusta.

Módulo auditoría

Figura 48

Módulo auditoría



Nota. visual estudio code (2026).

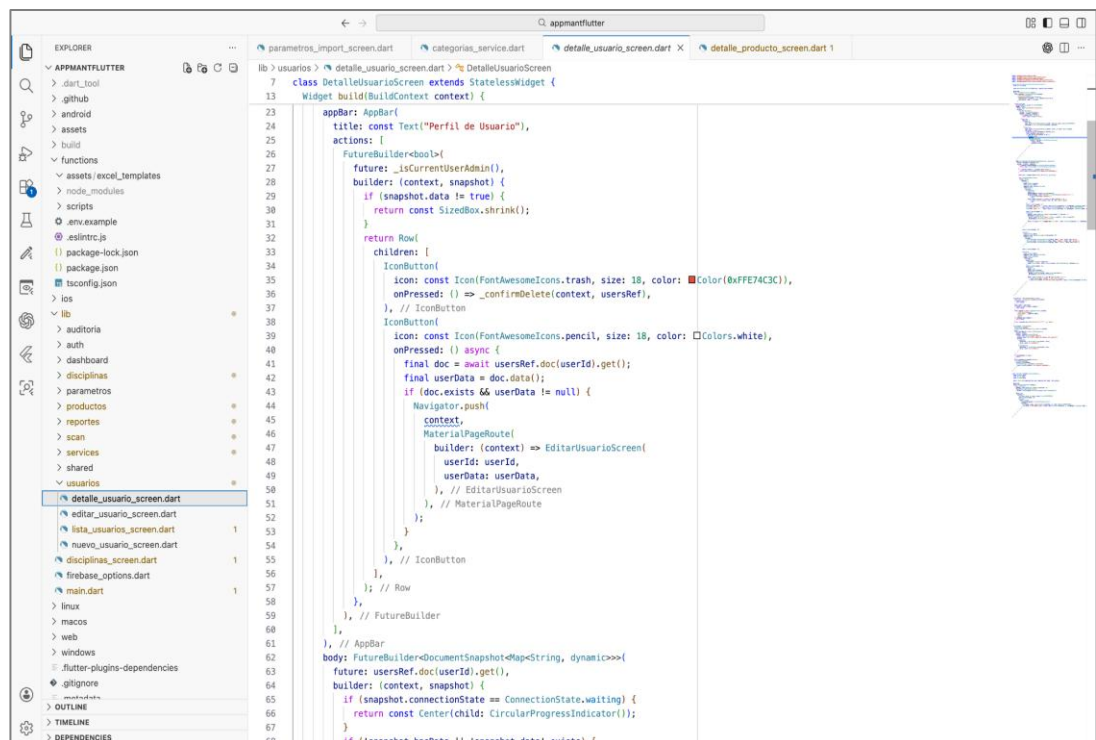
La figura muestra la implementación del módulo de Auditoría del aplicativo App-Mantenimiento, específicamente el archivo `auditoria_screen.dart`, desarrollado en el entorno

Visual Studio Code. Este componente formó parte del sistema encargado de registrar y visualizar las acciones realizadas por los usuarios dentro de la aplicación, permitiendo mantener un historial de eventos asociados al uso del sistema. En el código se observó la conexión con Cloud Firestore, desde donde se consultó la colección auditoria, en la cual se almacenaron los registros de las acciones ejecutadas por los usuarios. Mediante estas consultas, el sistema pudo filtrar, ordenar y mostrar cronológicamente los eventos registrados, permitiendo identificar quién realizó una acción, en qué momento se ejecutó y a qué elemento del sistema estuvo asociada. La implementación de este módulo permitió garantizar el control administrativo del sistema, facilitó el seguimiento de las operaciones realizadas, mantuvo la transparencia en el uso de la aplicación y proporcionó un historial verificable de eventos. De esta manera, el módulo de auditoría contribuyó a fortalecer la trazabilidad del sistema, permitiendo realizar análisis técnicos, control interno y supervisión de las actividades registradas dentro del aplicativo.

Módulo usuarios

Figura 49

Módulo usuarios



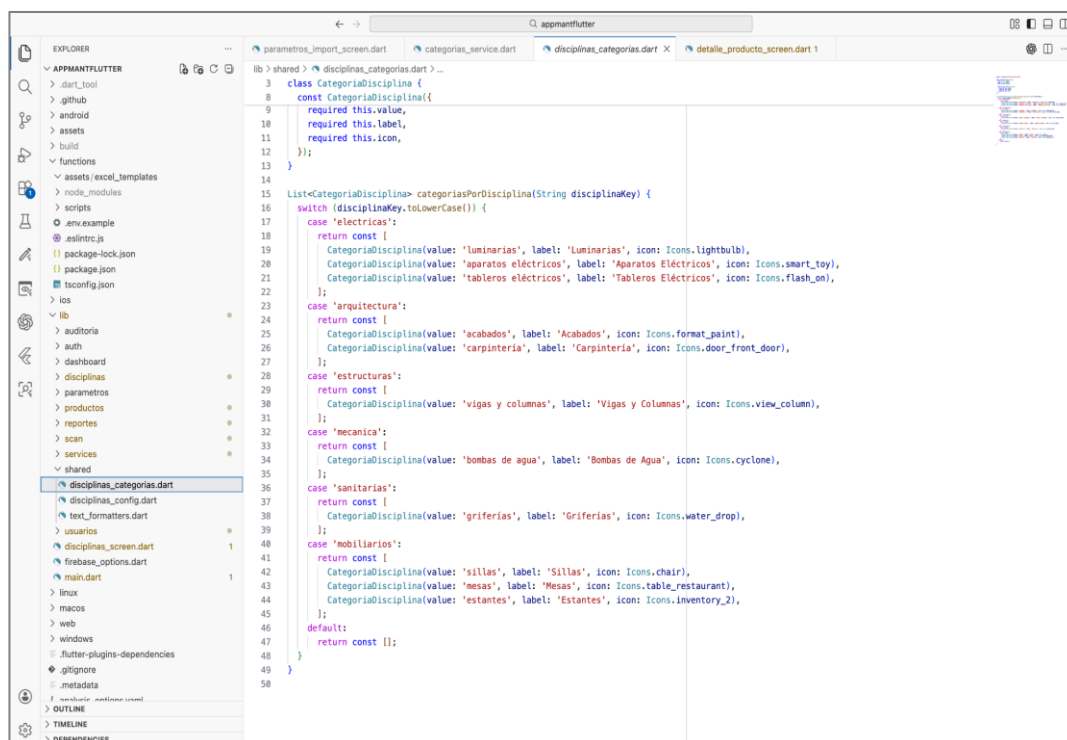
Nota. visual estudio code (2026).

La figura muestra la implementación del módulo de Usuarios del aplicativo App-Mantenimiento, específicamente el archivo `detalle_usuario_screen.dart`, desarrollado en el entorno Visual Studio Code. Este módulo se encargó de administrar la información extendida de los perfiles de usuario que interactúan con el sistema. En el código se observa que, aunque el proceso de autenticación principal se realizó mediante Firebase Authentication, el sistema almacenó información adicional en la colección usuarios de Cloud Firestore. Esta información complementaria permitió registrar datos del perfil del usuario, gestionar roles dentro del sistema, y habilitar funcionalidades para editar o actualizar la información del perfil. Asimismo, este módulo permitió implementar mecanismos de control de acceso, ya que el sistema puede verificar los roles y permisos asignados a cada usuario antes de habilitar determinadas funciones dentro de la aplicación. De esta manera, el módulo de usuarios contribuyó a organizar la gestión de perfiles, mejorar la administración del sistema y garantizar un acceso controlado a las diferentes funcionalidades del aplicativo

Módulo de componentes compartidos (shared/)

Figura 50

Estructura interna del módulo Shared.



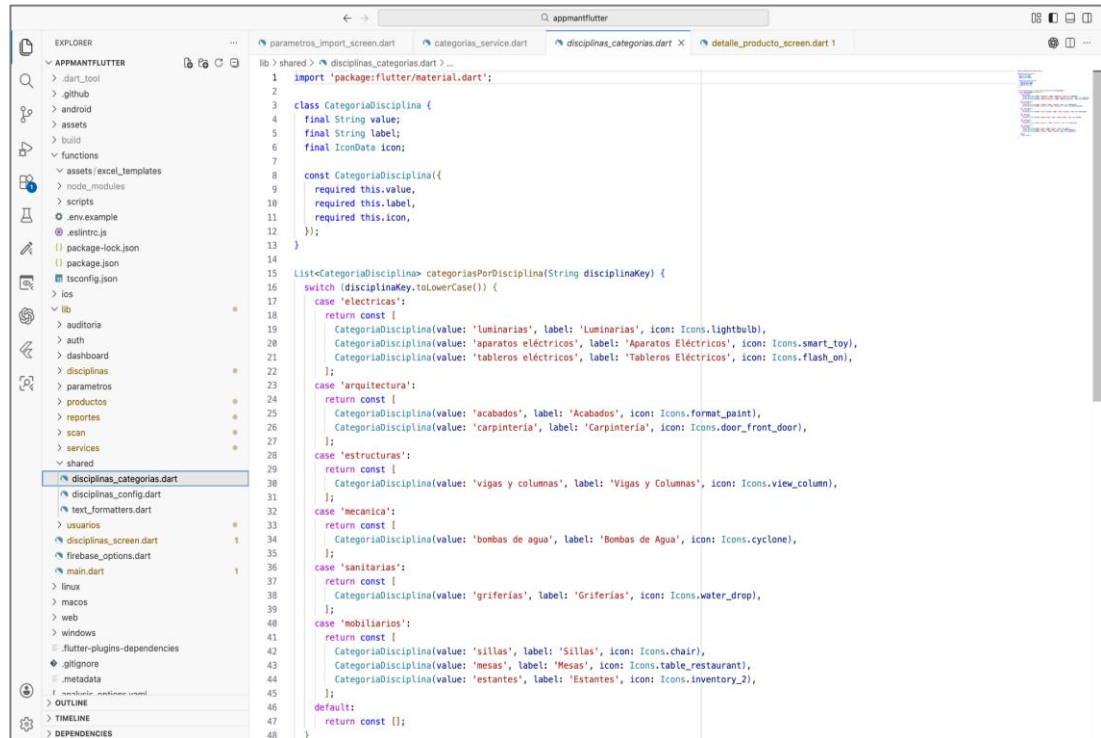
Nota. visual estudio code (2026).

La figura muestra la estructura interna del módulo shared/ del aplicativo App-Mantenimiento, visualizada en el entorno de desarrollo Visual Studio Code. Este módulo agrupa archivos como `disciplinas_categorias.dart`, `disciplinas_config.dart` y `text_formatters.dart`, los cuales contienen componentes y configuraciones utilizadas de manera común por diferentes partes del sistema. Dentro de este módulo se definieron estructuras de datos, configuraciones de disciplinas y utilidades para el manejo y validación de textos, permitiendo que estas funcionalidades puedan ser reutilizadas por múltiples pantallas y servicios del aplicativo. Por ejemplo, el archivo `disciplinas_categorias.dart` organizó las categorías asociadas a cada disciplina técnica, mientras que `text_formatters.dart` incluyó funciones para controlar y normalizar la entrada de datos en los formularios del sistema. La implementación de este módulo permitió centralizar elementos reutilizables y configuraciones globales, evitando la duplicación de código y facilitando la coherencia entre los distintos módulos del aplicativo. De esta manera, el módulo shared contribuyó a mejorar la organización del proyecto, la reutilización de componentes y la mantenibilidad.

Rol del Módulo Shared: La carpeta shared/ se utilizó para centralizar diversos componentes y configuraciones comunes del sistema que son utilizados por diferentes módulos del aplicativo. En este espacio se agruparon configuraciones generales del sistema, catálogos estáticos, formateadores de texto reutilizables y pequeñas funciones de lógica compartida, las cuales pueden ser utilizadas por múltiples pantallas y servicios de la aplicación.

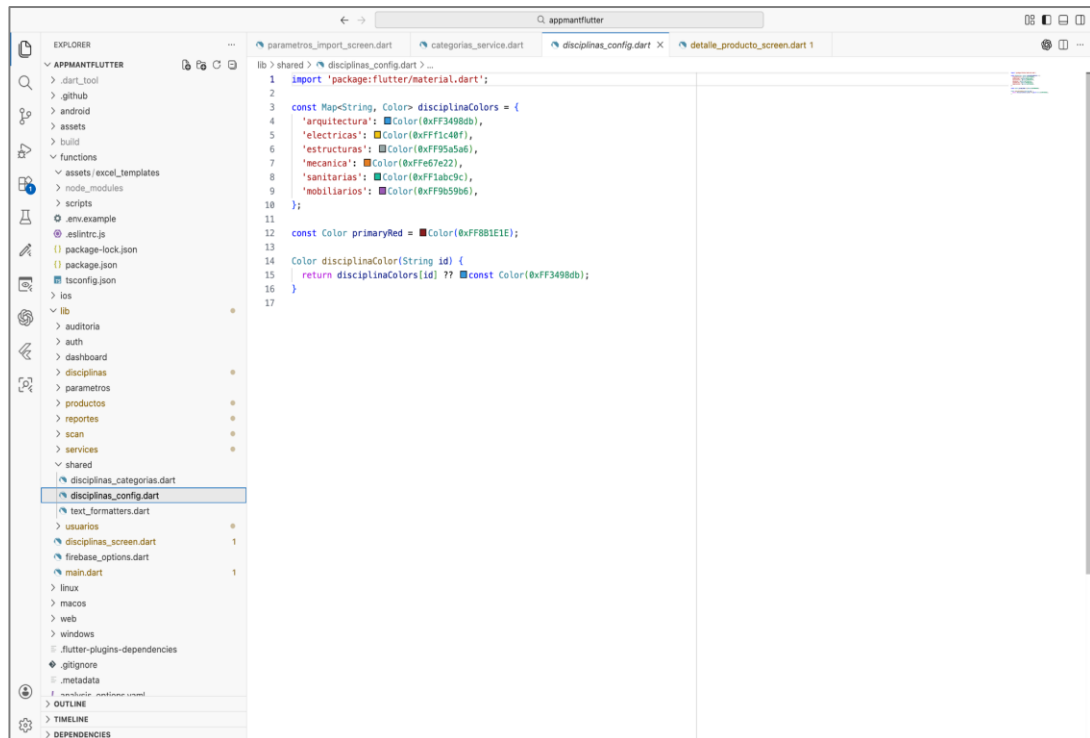
Análisis de Archivos: Se tiene lo siguiente:

Figura 51

Disciplinas_categorias.dart

Nota. visual estudio code (2026).

La figura muestra la implementación del archivo *disciplinas_categorias.dart*, ubicado en el módulo *shared/* del aplicativo App-Mantenimiento. En este componente se definió la clase *CategoriaDisciplina*, la cual estableció la estructura de las categorías mediante atributos como valor, etiqueta e ícono. Asimismo, se implementó el método *categoriasPorDisciplina()*, que permitió obtener las categorías correspondientes a cada disciplina técnica. Mediante esta configuración, el sistema organizó los activos y reportes de mantenimiento según su especialidad, como eléctricas, arquitectura, estructuras o mecánica. Este diseño permitió estandarizar la clasificación de los elementos, mantener una separación clara por disciplinas y asegurar coherencia con la organización del modelo BIM dentro del sistema.

Figura 52*Disciplinas_config.dart*

Nota. visual estudio code (2026).

La figura muestra el archivo `disciplinas_config.dart`, ubicado dentro del módulo `shared` del sistema. Este archivo tiene la función de centralizar la configuración relacionada con las disciplinas utilizadas en la aplicación. En él se definieron estructuras de datos que permiten asociar identificadores internos de disciplinas con atributos visuales, como colores representativos, los cuales fueron utilizados en las diferentes interfaces del sistema. Asimismo, el archivo estableció constantes globales, como colores primarios, y funciones que permitieron obtener automáticamente el color correspondiente a cada disciplina a partir de su identificador. Esta estrategia facilitó que todas las pantallas y componentes de la aplicación utilicen una misma fuente de configuración, garantizando consistencia visual y evitando la duplicación de código. En términos de arquitectura de software, esta implementación contribuyó a mejorar la mantenibilidad y escalabilidad del sistema, ya que cualquier modificación en la configuración de las disciplinas puede realizarse en un único punto, propagándose automáticamente al resto de la aplicación.

Figura 53*Text_formatters.dart*

```

lib > shared > text_formatters.dart > formatUpperCase
3 String formatTitleCase(String? value) {
4   if (value == null) return '';
5   final trimmed = value.trim();
6   if (trimmed.isEmpty) return '';
7   if (trimmed.contains('@') || trimmed.contains('/')) {
8     return trimmed;
9   }
10  final normalized = trimmed.replaceAll('_', ' ').replaceAll('-', ' ');
11  return normalized
12    .split(' ')
13    .map((word) {
14      if (word.isEmpty) return word;
15      final lower = word.toLowerCase();
16      return '${lower[0].toUpperCase()}${lower.substring(1)}';
17    })
18    .join(' ');
19 }
20
21 String formatUpperCase(String? value) {
22   if (value == null) return '';
23   final trimmed = value.trim();
24   if (trimmed.isEmpty) return '';
25   return trimmed.toUpperCase();
26 }
27
28 String formatSubcategoriaDisplay(String? value) {
29   return formatTitleCase(value);
30 }
31
32 String normalizeCategoriaValue(String value) {
33   var text = value.trim().toLowerCase();
34   if (text.isEmpty) return '';
35   const replacements = {
36     'á': 'a',
37     'é': 'e',
38     'í': 'i',
39     'ó': 'o',
40     'u': 'u',
41     'ü': 'u',
42     'ñ': 'n',
43   };
44   final buffer = StringBuffer();
45   for (final rune in text.runes) {
46     final char = String.fromCharCode(rune);
47     buffer.write(replacements[char] ?? char);
48   }
49   text = buffer.toString();
50   text = text.replaceAll(RegExp('[a-z0-9]*'), ' ');

```

Nota. visual estudio code (2026).

La figura muestra la implementación del archivo `text_formatters.dart`, ubicado dentro del módulo `shared/` del aplicativo App-Mantenimiento, visualizado en el entorno de desarrollo Visual Studio Code. Este componente incluyó funciones destinadas a formatear, validar y normalizar los datos ingresados por el usuario en los formularios del sistema. En el código se observaron métodos como `formatTitleCase`, `formatUpperCase` y `normalizeCategoriaValue`, los cuales permitieron estandarizar la escritura de los textos, convertir caracteres a formatos específicos y limpiar caracteres especiales. Estas funciones se aplicaron durante el ingreso de datos para asegurar que la información almacenada mantenga un formato consistente. La implementación de este archivo contribuyó al control de calidad de los datos desde la interfaz de usuario, permitiendo validar y normalizar la información antes de ser almacenada en la base de datos, lo que ayudó a mantener coherencia, integridad y uniformidad en los registros del sistema.

d. Fragmento de código relevantes

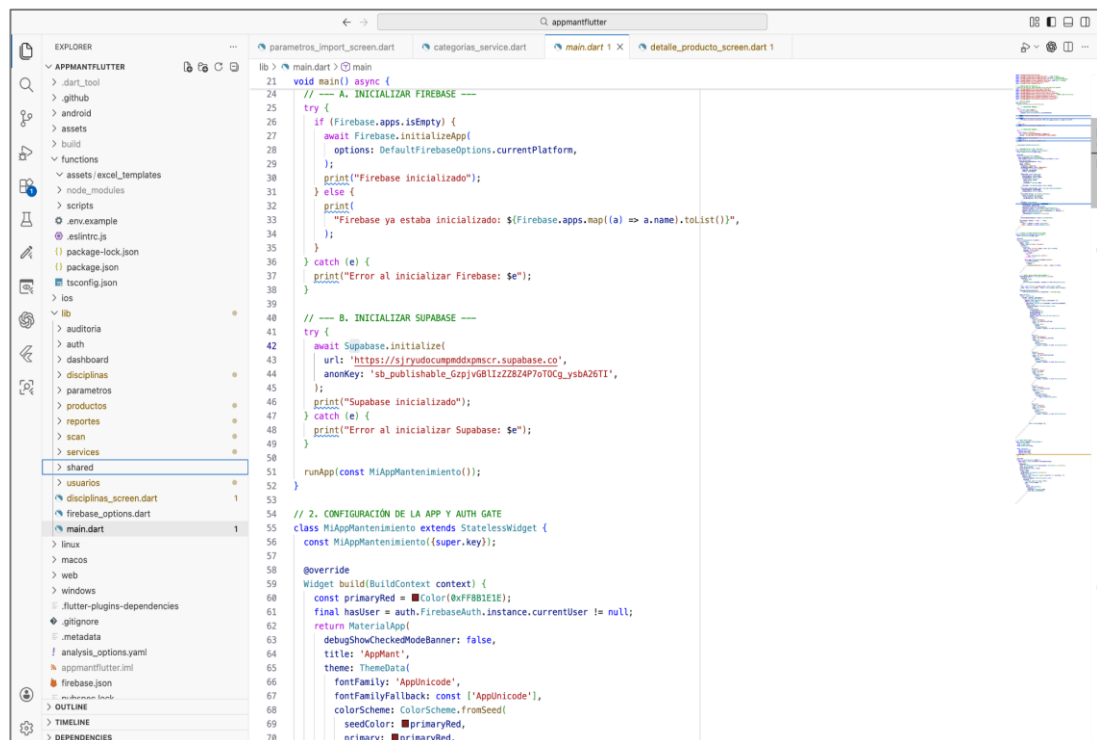
El presente apartado expone los fragmentos de código más representativos del desarrollo del aplicativo App-Mantenimiento, los cuales permitieron a evidenciar la

implementación técnica de los principales procesos funcionales que conforman el sistema. En esta sección se incluyeron capturas del código fuente desarrollado en Flutter, con la finalidad de sustentar la correcta integración entre la aplicación móvil, los servicios en la nube y la base de datos estructurada que soporta el funcionamiento de la plataforma. Los fragmentos seleccionados corresponden a la inicialización del servicio Firebase, el proceso de autenticación o inicio de sesión de los usuarios, las operaciones CRUD relacionadas con la gestión de activos, la implementación del escáner de códigos QR para la identificación de equipos, así como el proceso de carga de imágenes en almacenamiento externo. Cada uno de estos componentes constituyó una parte fundamental dentro de la arquitectura del sistema, ya que permitieron garantizar la interacción eficiente entre la interfaz de usuario, la lógica de negocio y los servicios de almacenamiento y procesamiento.

Inicialización de Firebase

Figura 54

Inicialización de Firebase en el archivo main.dart



Nota. visual estudio code (2026).

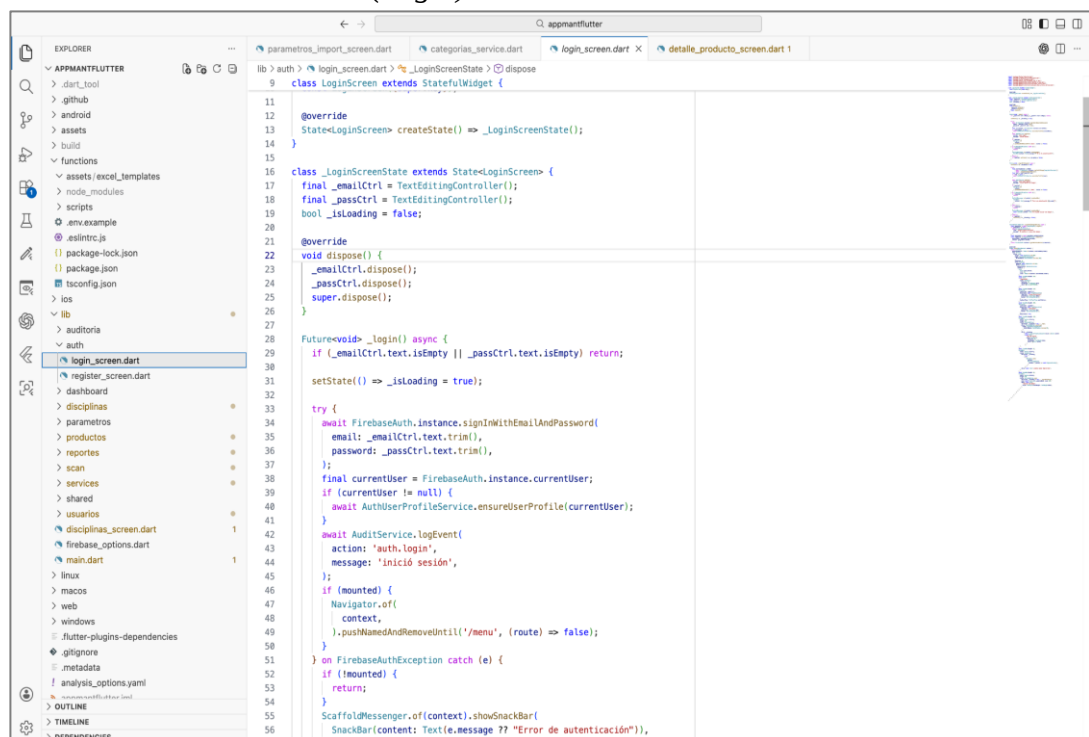
La figura muestra la inicialización de Firebase, se realizó en el archivo principal main.dart, el cual constituyó el punto de entrada de la aplicación desarrollada en Flutter. En este proceso se ejecutó el método void main(), donde primero se garantizó la correcta

inicialización del entorno mediante `WidgetsFlutterBinding.ensureInitialized()`. Posteriormente, se estableció la conexión con el proyecto de Firebase mediante la función `Firebase.initializeApp()`, utilizando la configuración `DefaultFirebaseOptions.currentPlatform`, que contiene los parámetros específicos del proyecto para cada plataforma. Finalmente, una vez completada la configuración, se ejecutó el método `runApp()`, permitiendo iniciar la aplicación con los servicios de Firebase correctamente habilitados, lo cual aseguró la disponibilidad de los servicios de autenticación, base de datos y almacenamiento desde el inicio de la ejecución del sistema.

Proceso de Inicio de Sesión (Login)

Figura 55

Proceso de Inicio de Sesión (Login)



Nota. visual estudio code (2026).

La figura muestra el proceso de inicio de sesión, donde se implementó en el módulo auth, específicamente en el archivo `login_screen.dart`, donde se gestionó la autenticación de los usuarios mediante Firebase Authentication. En este procedimiento se validaron previamente los campos del formulario para verificar que el usuario ingresara correctamente su correo electrónico y contraseña. Posteriormente, se ejecutó el método de autenticación

signInWithEmailAndPassword, el cual permitió verificar las credenciales ingresadas con los registros almacenados en Firebase. En caso de que la autenticación fuera exitosa, el sistema obtuvo la información del usuario autenticado y permitió el acceso a la interfaz principal de la aplicación. Asimismo, se utilizó una estructura try/catch para gestionar posibles errores durante el proceso de autenticación, permitiendo mostrar mensajes de error al usuario en caso de credenciales incorrectas o fallos en la conexión. De esta manera, se garantizó un proceso de inicio de sesión seguro y controlado dentro del sistema.

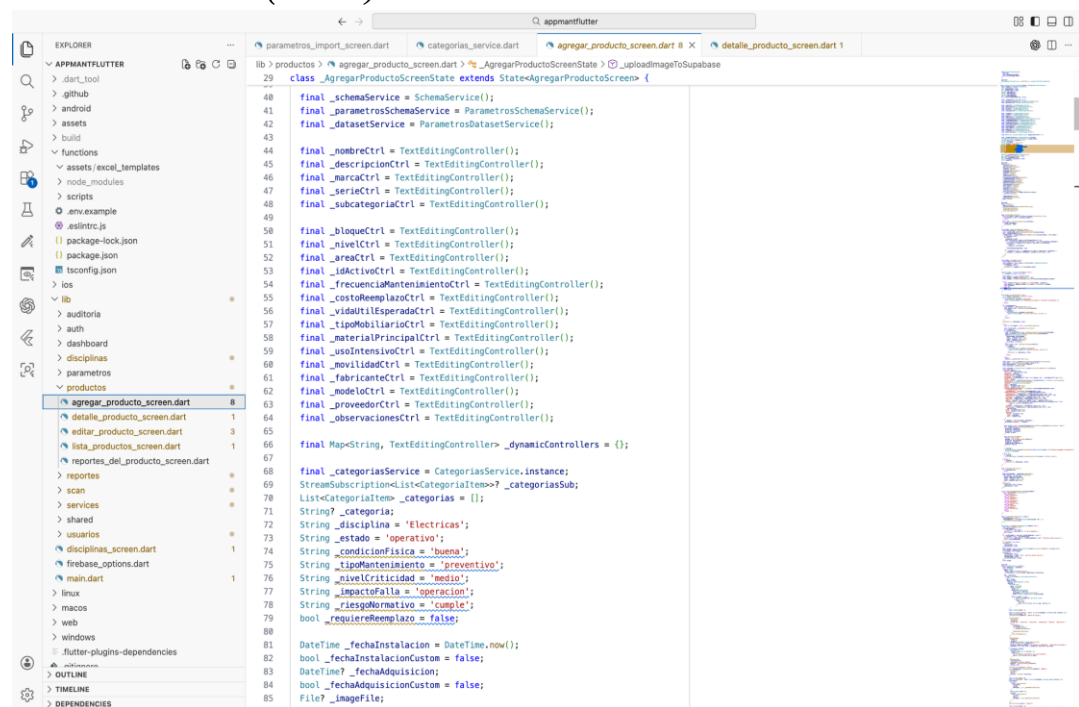
Operaciones CRUD de Activos

El módulo de productos implementó las operaciones CRUD (Create, Read, Update y Delete), las cuales permitieron gestionar de manera integral los activos importados desde el modelo BIM dentro del sistema. A través de estas operaciones se posibilitó registrar nuevos activos, consultar la información almacenada, actualizar los datos cuando fue necesario y eliminar registros que ya no resultaban requeridos.

Creación de Activo (Create): Este fragmento demostró cómo se registra un nuevo activo dentro de la base de datos.

Figura 56

Creación de Activo (Create)



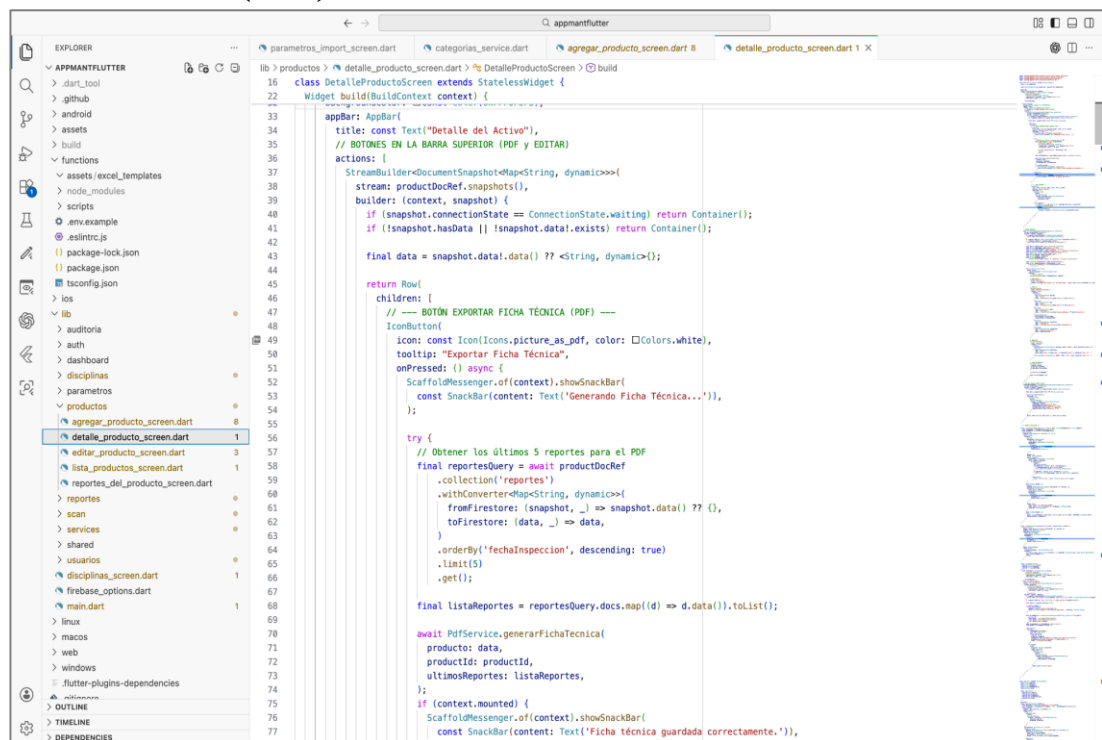
Nota. visual estudio code (2026).

La figura muestra la creación de activos, donde se implementó dentro del módulo de productos mediante un formulario que permitió registrar la información correspondiente a cada elemento importado desde el modelo BIM. En este proceso se capturaron diversos datos del activo, tales como nombre, descripción, ubicación, disciplina, nivel, bloque y otras características técnicas necesarias para su identificación dentro del sistema. Posteriormente, la información ingresada fue validada para verificar que los campos obligatorios estuvieran completos y que el identificador del activo no estuviera duplicado. Una vez superada esta validación, el sistema procedió a registrar el nuevo activo en la base de datos Firestore, almacenando los datos dentro de la colección correspondiente a los productos o activos. Asimismo, el procedimiento incluyó el manejo de errores mediante estructuras de control que permitieron notificar al usuario en caso de inconsistencias en la información o fallos durante el registro. De esta manera, se garantizó el almacenamiento correcto y seguro de los activos dentro del sistema.

Lectura de Activo (Read): Este fragmento evidenció la recuperación dinámica de información almacenada en la base de datos.

Figura 57

Lectura de Activo (Read)



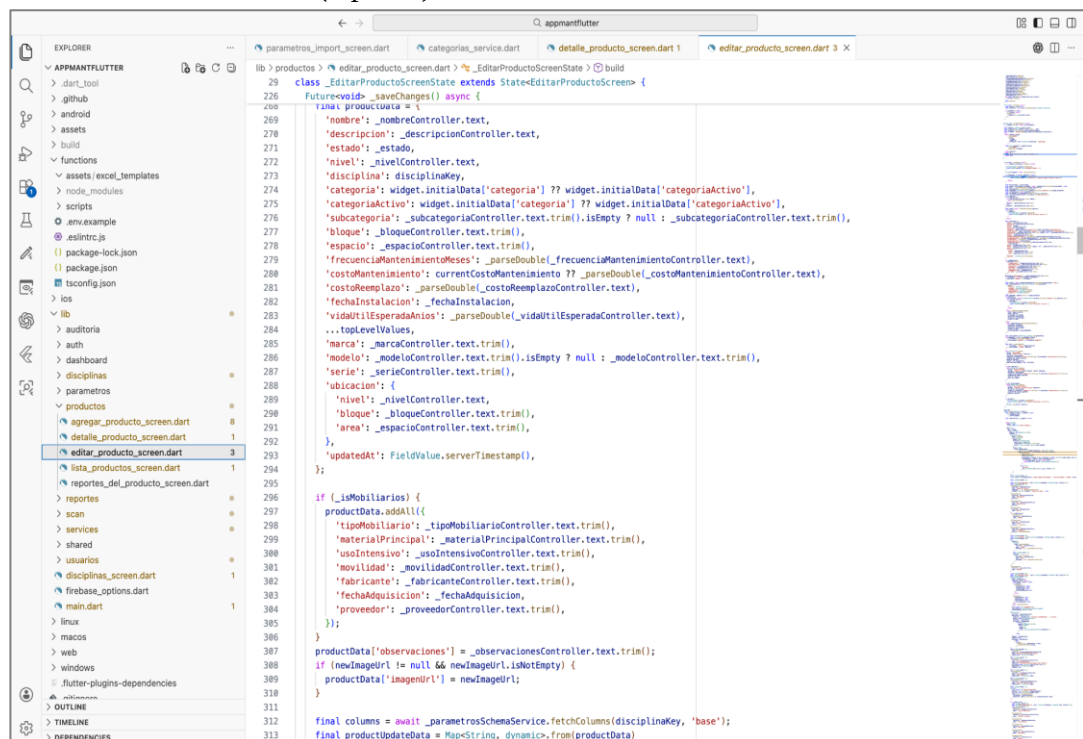
Nota. visual estudio code (2026).

La figura muestra la lectura de activos, que se realizó mediante una consulta a la base de datos Firestore, utilizando el identificador único del producto para acceder al documento correspondiente dentro de la colección productos. Para ello, el sistema empleó la referencia `.doc(productId)`, lo que permitió localizar de manera específica el registro asociado al activo seleccionado. Posteriormente, la información almacenada se recuperó de forma dinámica mediante el uso de `snapshots()`, lo cual permitió obtener y visualizar los datos actualizados del activo en tiempo real dentro de la interfaz de la aplicación. A partir de esta consulta, el sistema extrajo diversos campos del registro, tales como nombre, categoría, disciplina, ubicación, descripción y otros atributos técnicos asociados al activo. De esta manera, el sistema permitió visualizar la información detallada del activo seleccionado, garantizando la recuperación eficiente y actualizada de los datos almacenados en la base de datos.

Actualización de Activo (Update): Este bloque demuestra cómo el sistema permitió modificar información existente sin necesidad de eliminar el documento completo.

Figura 58

Actualización de Activo (Update)



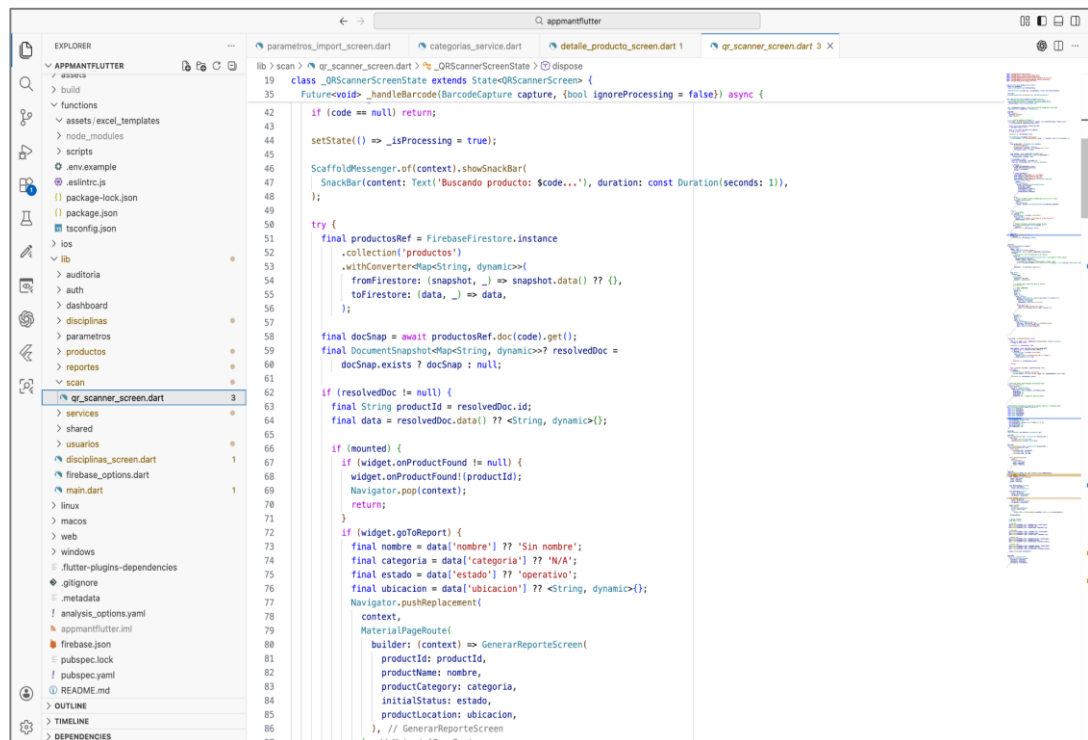
Nota. visual estudio code (2026).

La figura muestra la actualización de activos, donde se implementó mediante un formulario que permitió modificar la información previamente registrada en la base de datos. En este proceso, el sistema recuperó los datos existentes del activo y permitió editar diversos campos, tales como nombre, descripción, ubicación, proveedor, características técnicas y observaciones. Una vez realizados los cambios, el sistema procesó la información actualizada y ejecutó el método correspondiente para modificar los datos almacenados en Firestore, utilizando el identificador único del activo para localizar el documento dentro de la colección productos. Este procedimiento permitió actualizar únicamente los campos modificados sin necesidad de eliminar el documento completo, garantizando así la integridad de la información y permitiendo mantener actualizados los registros de los activos dentro del sistema.

Implementación del Escáner QR

Figura 59

Implementación del Escáner QR



Nota. visual estudio code (2026).

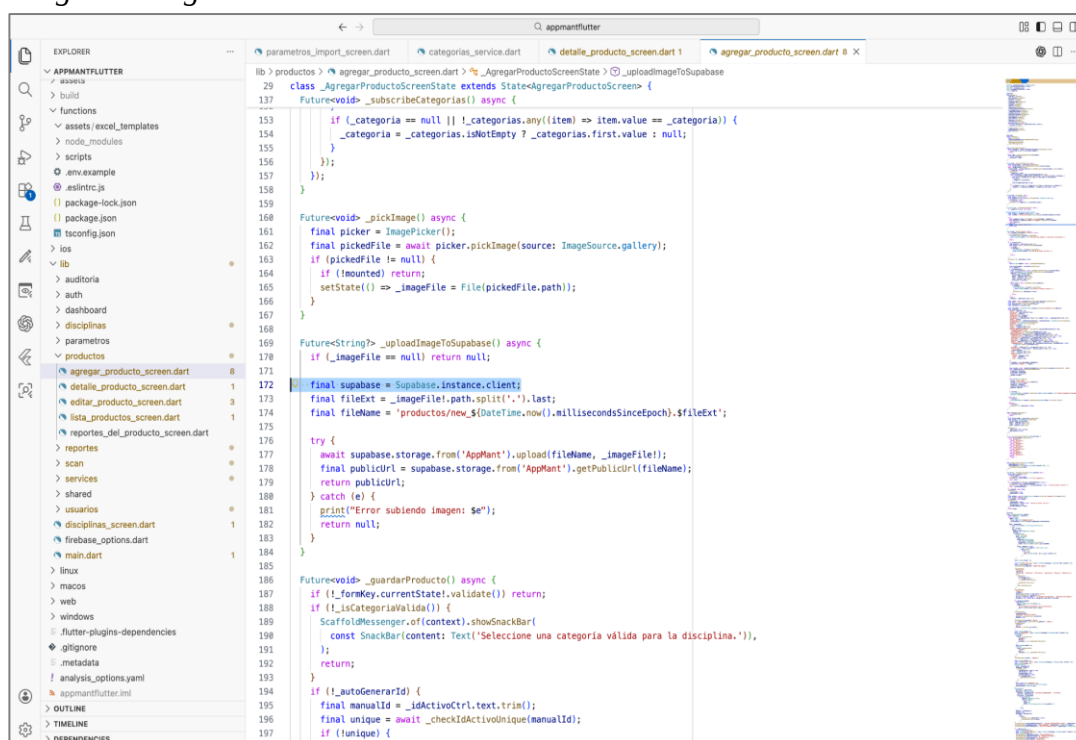
La figura muestra la implementación del escáner QR se desarrolló dentro del módulo scan, permitiendo la identificación rápida de activos mediante el uso de la cámara del dispositivo móvil. En este proceso, el sistema capturó el código QR escaneado y extrajo el

valor contenido en el mismo, el cual correspondía al identificador del activo registrado en la base de datos. Posteriormente, el código obtenido fue utilizado para realizar una consulta en Firestore, específicamente en la colección de productos, con el objetivo de verificar si el activo se encontraba registrado en el sistema. En caso de que el registro existiera, la aplicación redirigió automáticamente al usuario hacia la pantalla de detalle del activo, donde se mostraba la información completa del equipo. De esta manera, el sistema integró el uso del hardware del dispositivo (cámara), el procesamiento del código QR y la consulta en tiempo real a la base de datos, facilitando la localización y consulta rápida de los activos durante las actividades de mantenimiento en campo.

Carga de Imágenes en Almacenamiento Externo

Figura 60

Carga de Imágenes en Almacenamiento Externo



Nota. visual estudio code (2026).

La figura muestra la carga de imágenes de los activos se implementó mediante el uso de Supabase Storage, lo cual permitió almacenar archivos multimedia de forma externa a la base de datos principal. En este proceso, el sistema permitió seleccionar una imagen desde la galería del dispositivo y posteriormente procesarla para su envío al servicio de

almacenamiento en la nube. Una vez seleccionada la imagen, se inicializó el cliente de Supabase y se ejecutó el método `upload(...)`, mediante el cual el archivo fue cargado al contenedor de almacenamiento correspondiente. Posteriormente, el sistema obtuvo la URL pública del archivo utilizando el método `getPublicUrl(...)`, lo que permitió acceder a la imagen desde la aplicación. Finalmente, la URL generada fue asociada al registro del activo dentro de la base de datos Firestore, permitiendo que la imagen quedara vinculada al equipo registrado en el sistema. De esta manera, se garantizó una correcta gestión del almacenamiento de archivos y su integración con la información estructurada del aplicativo.

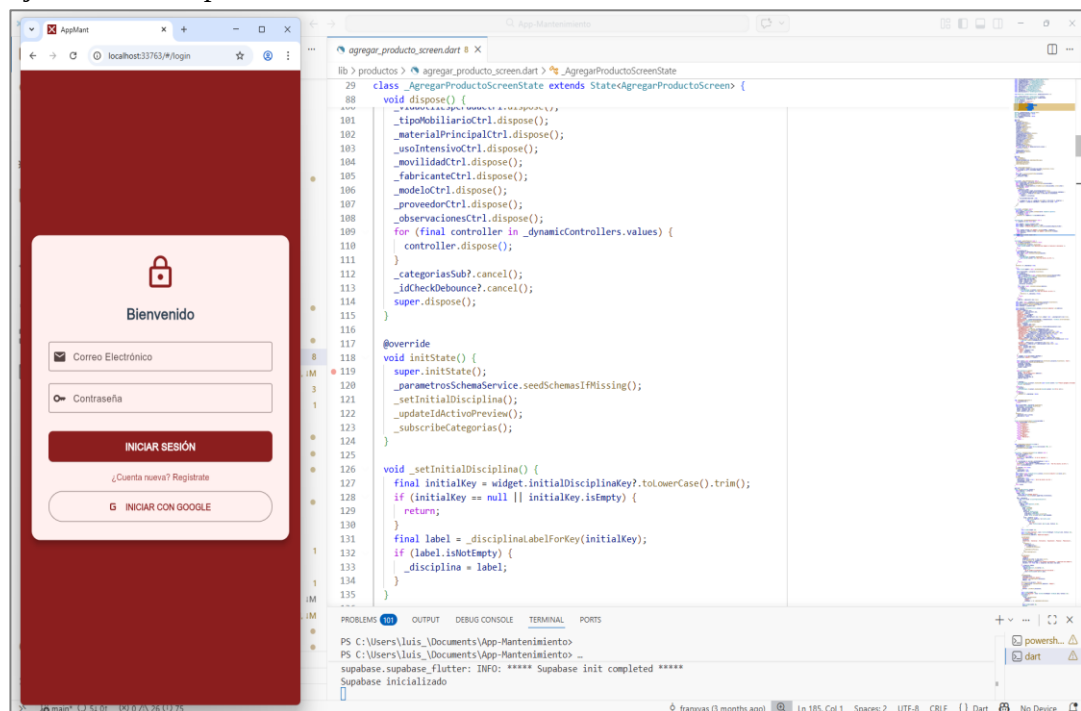
e. Ejecución de la aplicación

Se describe el procedimiento técnico necesario para ejecutar el aplicativo App-Mantenimiento en los entornos soportados por el sistema. La aplicación fue desarrollada utilizando Flutter, permitiendo su despliegue multiplataforma en entorno Web y dispositivos Android.

Ejecución en Entorno Web

Figura 61

Ejecución del aplicativo en entorno Web



Nota. visual estudio code (2026).

La figura muestra la ejecución del aplicativo en entorno web, donde se realizó utilizando las herramientas de desarrollo de Flutter. Para ello, inicialmente se abrió una terminal en la raíz del proyecto con el fin de ejecutar los comandos necesarios para la compilación y despliegue de la aplicación. Posteriormente, se verificaron los dispositivos disponibles mediante el comando `flutter devices`, lo cual permitió identificar los entornos en los que la aplicación podía ejecutarse. Una vez confirmado que el navegador Chrome se encontraba disponible como dispositivo de ejecución, se procedió a iniciar la aplicación mediante el comando `flutter run -d chrome`. Como resultado, el sistema fue compilado y ejecutado en el navegador web, permitiendo visualizar la interfaz del aplicativo y verificar su funcionamiento, incluyendo el acceso al módulo de inicio de sesión y la correcta conexión con los servicios en la nube utilizados por el sistema.

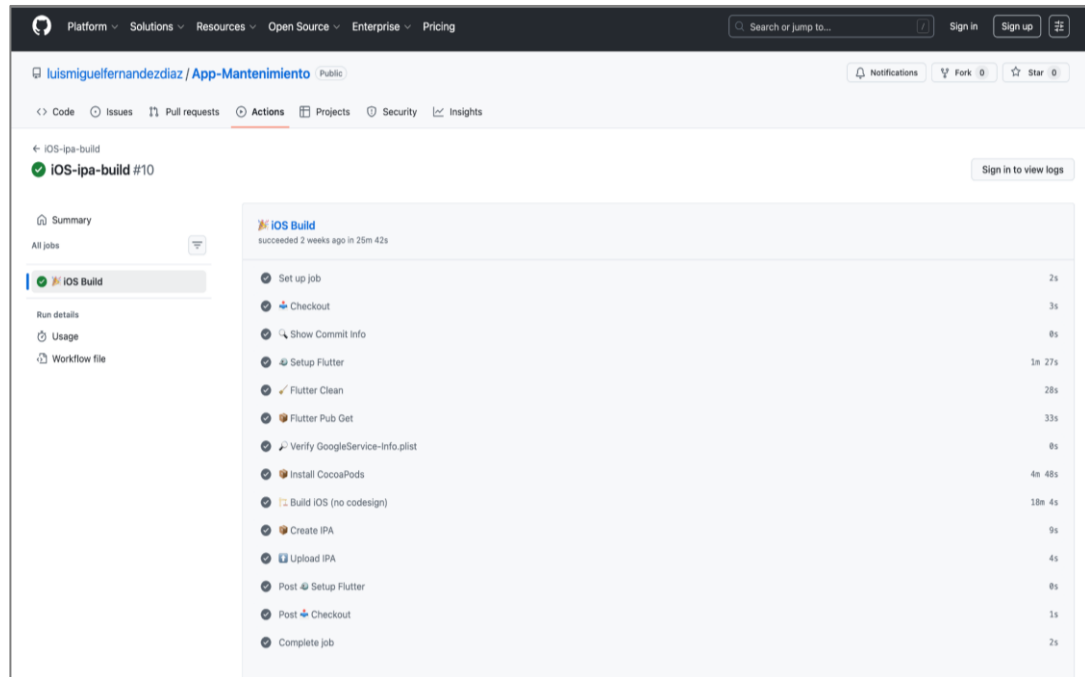
Generación e Instalación del Aplicativo para iOS

Durante la fase final del desarrollo del aplicativo App-Mantenimiento, la generación del paquete instalable para dispositivos móviles se realizó mediante un proceso automatizado de compilación utilizando GitHub Actions, permitiendo construir la aplicación para el sistema operativo iOS y generar un archivo instalable en formato IPA (iOS App Store Package). Este proceso automatizado se ejecutó en un entorno virtual basado en macOS, requisito indispensable para compilar aplicaciones destinadas al ecosistema Apple. El flujo de compilación se encontró definido mediante un archivo de configuración YAML dentro del repositorio del proyecto, el cual describe cada una de las etapas necesarias para preparar el entorno, compilar el aplicativo Flutter y generar el paquete final de distribución. El proceso de compilación se activó manualmente mediante el evento `workflow_dispatch`, lo que permitió ejecutar el flujo de construcción bajo demanda desde la plataforma GitHub.

Preparación del entorno de compilación: En la primera etapa del flujo de trabajo se descargó el código fuente del repositorio mediante el proceso de checkout, lo cual permitió acceder a la versión actual del proyecto. Posteriormente, se mostró información relevante del commit utilizado para la compilación, incluyendo la rama activa y el identificador del commit. Este procedimiento permitió mantener la trazabilidad sobre la versión exacta del código empleada para generar el aplicativo. Seguidamente, se configuró el entorno de desarrollo de Flutter mediante la instalación del SDK en su versión estable, asegurando que todas las herramientas necesarias estuvieran disponibles para el proceso de compilación.

Figura 62

Ejecución del flujo de compilación automatizado en GitHub Actions.



Nota. visual estudio code (2026).

La compilación del aplicativo se realizó mediante un flujo de integración continua utilizando GitHub Actions, lo cual permitió automatizar el proceso de construcción del proyecto. En este flujo de trabajo, inicialmente se descargó el código fuente del repositorio y se verificó la información del commit correspondiente a la versión utilizada para la compilación. Posteriormente, se configuró el entorno de desarrollo mediante la instalación del SDK de Flutter y se ejecutaron procesos de limpieza y descarga de dependencias del proyecto. Asimismo, se verificó la configuración del archivo GoogleService-Info.plist, necesario para la integración con los servicios de Firebase en la plataforma iOS.

Preparación del proyecto Flutter: Antes de iniciar el proceso de compilación del aplicativo, se ejecutaron comandos destinados a preparar el proyecto y garantizar que todas las dependencias estuvieran correctamente instaladas. En primer lugar, se ejecutó el comando flutter clean, el cual permitió eliminar archivos temporales y compilaciones previas, asegurando que el proceso de construcción se realizara desde un estado limpio. Posteriormente, se ejecutó el comando flutter pub get, mediante el cual se descargaron e instalaron todas las dependencias definidas en el archivo pubspec.yaml. Este procedimiento permitió disponer de las librerías necesarias para el correcto funcionamiento del aplicativo durante el proceso de compilación.

Verificación de configuración de Firebase: Durante el flujo de compilación se verificó la existencia del archivo de configuración de Firebase correspondiente a la plataforma iOS, denominado GoogleService-Info.plist. Este archivo contiene la información necesaria para establecer la conexión entre la aplicación y los servicios de Firebase, tales como autenticación, base de datos y almacenamiento. Asimismo, el sistema validó que dicho archivo se encontrara correctamente ubicado dentro del directorio ios/Runner/ del proyecto. En caso de que el archivo no estuviera presente, el proceso de compilación se detenía automáticamente, con el fin de evitar errores durante la construcción y ejecución del aplicativo.

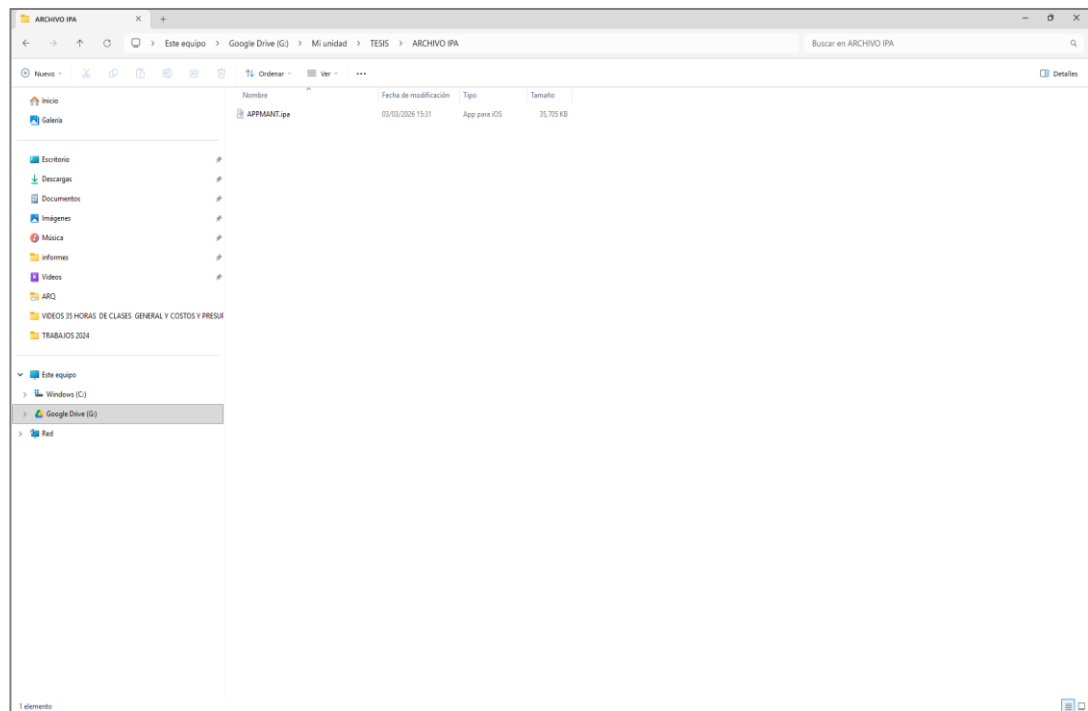
Instalación de dependencias del proyecto iOS: Una vez verificada la configuración del proyecto, se procedió a instalar las dependencias específicas de iOS mediante el gestor de paquetes CocoaPods. Para ello, se ejecutó el comando `pod install --repo-update`, el cual permitió descargar e instalar las librerías necesarias para el proyecto. Este procedimiento facilitó la integración del aplicativo desarrollado en Flutter con las APIs nativas del sistema operativo iOS, asegurando que todos los componentes y dependencias requeridas estuvieran correctamente configurados para el proceso de compilación.

Compilación del aplicativo: Posteriormente se ejecutó la compilación del proyecto Flutter en modo release, con el propósito de generar una versión optimizada del aplicativo destinada a su distribución. Para ello, se utilizó el comando `flutter build ios --release --no-codesign`, el cual permitió construir el proyecto sin realizar el proceso de firma digital. El parámetro `--no-codesign` se empleó debido a que la compilación se realizó dentro de un entorno de integración continua, donde la firma digital puede gestionarse en etapas posteriores del flujo de despliegue. Durante esta etapa, Flutter generó el archivo `Runner.app`, el cual corresponde al paquete compilado de la aplicación y contiene los recursos y binarios necesarios para su ejecución en dispositivos con sistema operativo iOS.

Generación del archivo IPA: Una vez compilado el aplicativo, se procedió a generar el archivo instalable en formato IPA. Este proceso consistió en organizar el paquete de la aplicación dentro de una carpeta denominada Payload y posteriormente comprimir el contenido para generar el archivo final.

Figura 63

Este archivo constituye el paquete instalable del aplicativo para dispositivos iOS.



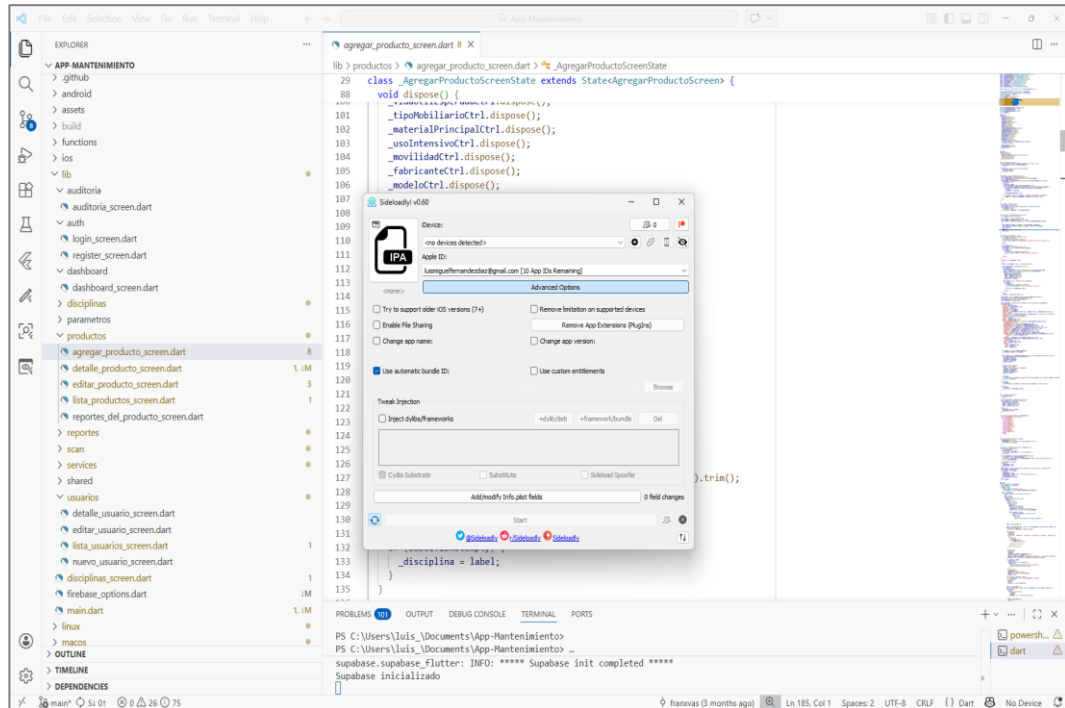
Nota. visual estudio code (2026).

La figura muestra el proceso de compilación automatizada se generó el archivo FlutterIpaExport.ipa, el cual corresponde al paquete instalable del aplicativo para dispositivos con sistema operativo iOS. Este archivo contiene el aplicativo compilado junto con todos los recursos y dependencias necesarios para su ejecución. El archivo IPA constituyó el formato utilizado por Apple para la distribución e instalación de aplicaciones en dispositivos iPhone o iPad. Su generación confirma que el proceso de compilación se ejecutó correctamente dentro del flujo automatizado, permitiendo disponer del aplicativo listo para su instalación, pruebas o distribución en la plataforma iOS.

Instalación del aplicativo mediante Sideloadly: En este procedimiento, primero se conectó el dispositivo iOS al equipo de desarrollo mediante un cable USB. Posteriormente, se abrió la herramienta Sideloadly y se seleccionó el archivo IPA generado durante el proceso de compilación. A continuación, se ingresaron las credenciales del Apple ID, las cuales permitieron realizar una firma temporal de la aplicación para su instalación. Finalmente, se ejecutó el proceso de instalación, permitiendo que el aplicativo quedara disponible en el dispositivo para su correspondiente evaluación y pruebas de funcionamiento.

Figura 64

Instalación del aplicativo en dispositivo iOS utilizando Sideloadly.



Nota. visual estudio code (2026).

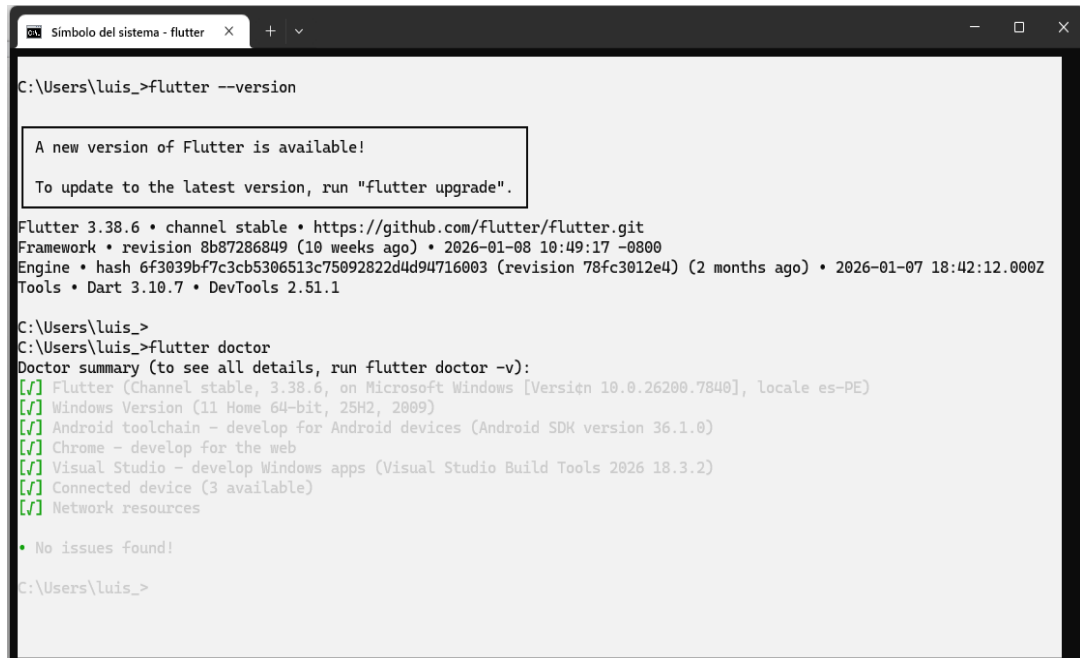
La figura muestra, la instalación del aplicativo en un dispositivo iOS se realizó utilizando la herramienta Sideloadly, la cual permitió cargar el archivo IPA generado durante el proceso de compilación directamente en el dispositivo móvil. Para ello, el dispositivo fue conectado al equipo de desarrollo mediante un cable USB, y posteriormente se seleccionó el archivo del aplicativo dentro de la interfaz de la herramienta. Durante el proceso, se ingresaron las credenciales del Apple ID, las cuales permitieron realizar la firma temporal de la aplicación necesaria para su instalación en el dispositivo. Una vez completado este procedimiento, el aplicativo fue instalado correctamente en el dispositivo iOS. ste método permitió realizar pruebas funcionales en un entorno real, validando el correcto funcionamiento de diferentes módulos del sistema, tales como la navegación entre pantallas, el escaneo de códigos QR, el registro de activos y la gestión de reportes dentro de la aplicación.

Verificación del Entorno de Desarrollo

Durante el desarrollo se utilizó el comando de diagnóstico para verificar la correcta configuración del entorno. flutter doctor

Figura 65

Estado del entorno de desarrollo mediante flutter doctor.



```
C:\Users\luis_>flutter --version

A new version of Flutter is available!

To update to the latest version, run "flutter upgrade".

Flutter 3.38.6 • channel stable • https://github.com/flutter/flutter.git
Framework • revision 8b87286849 (10 weeks ago) • 2026-01-08 10:49:17 -0800
Engine • hash 6f3039bf7c3cb5306513c75092822d4d94716003 (revision 78fc3012e4) (2 months ago) • 2026-01-07 18:42:12.000Z
Tools • Dart 3.10.7 • DevTools 2.51.1

C:\Users\luis_>
C:\Users\luis_>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.38.6, on Microsoft Windows [Versión 10.0.26200.7840], locale es-PE)
[✓] Windows Version (11 Home 64-bit, 25H2, 2009)
[✓] Android toolchain - develop for Android devices (Android SDK version 36.1.0)
[✓] Chrome - develop for the web
[✓] Visual Studio - develop Windows apps (Visual Studio Build Tools 2026 18.3.2)
[✓] Connected device (3 available)
[✓] Network resources

• No issues found!

C:\Users\luis_>
```

Nota. visual estudio code (2026).

Durante el proceso de desarrollo del aplicativo se utilizó el comando flutter doctor con el objetivo de verificar el estado y la correcta configuración del entorno de desarrollo. Este comando permitió realizar un diagnóstico automático de los componentes necesarios para trabajar con el framework Flutter. Como resultado de la ejecución, el sistema confirmó que las principales herramientas requeridas se encontraban correctamente instaladas y configuradas, incluyendo el SDK de Flutter, el Android SDK, el soporte para desarrollo web mediante Chrome, así como las herramientas de compilación de Visual Studio para aplicaciones en Windows. Asimismo, el diagnóstico indicó la disponibilidad de dispositivos conectados para la ejecución y prueba del aplicativo. Finalmente, el resultado “No issues found” confirmó que no se detectaron errores en la configuración del entorno, garantizando así que el sistema estaba preparado para el desarrollo, compilación y ejecución de la aplicación.

2.3.4.3. FASE 3: Modelado BIM de la Institución Educativa

a. Levantamiento arquitectónico.

Figura 66

Toma de medidas de la infraestructura



Nota. Elaboración propia.

La figura muestra la toma de medidas del bloque I, donde se tomaron medidas internas y externas, con el fin de poder realizar planos, seguido del modelado.

Figura 67

Segundo piso



Nota. Elaboración propia.

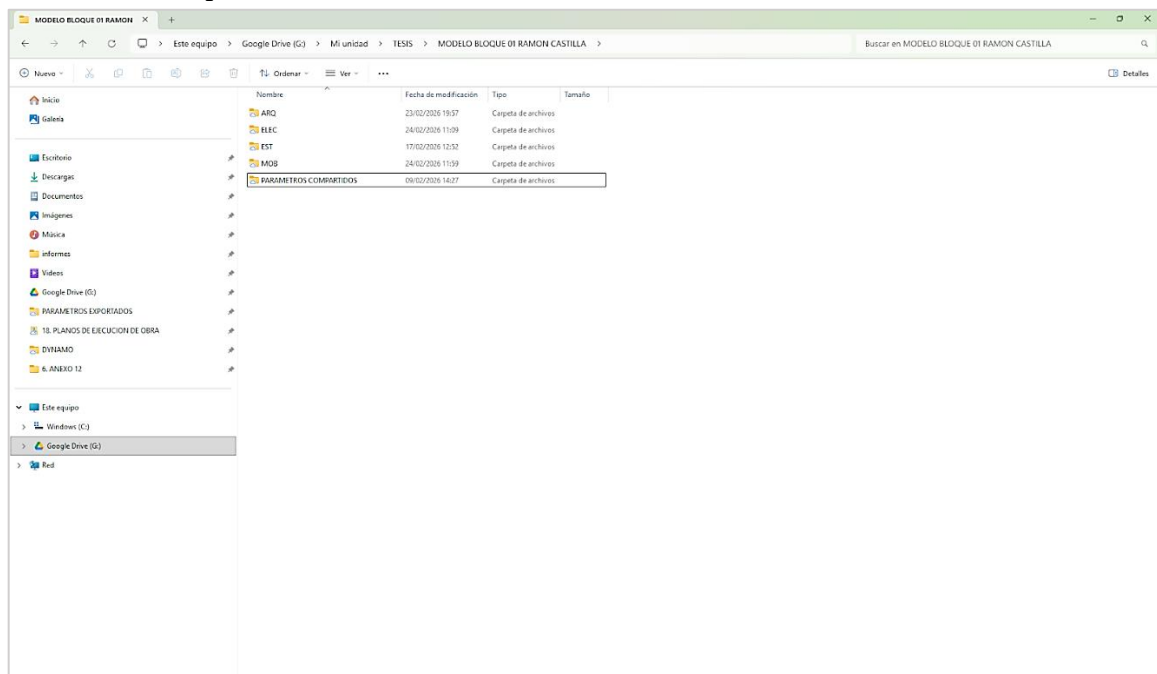
La figura muestra la toma de datos del ambiente del primer piso, como el tipo de mobiliario, puertas, ventanas, pizarras, Tv, interruptores, tomacorrientes, focos y condiciones generales de la infraestructura.

b. Modelado de estructura 3D en software REVIT

Se usó el software Revit 2025 donde se modeló de manera independiente cada disciplina tomando como base el modelo de estructuras del bloque I.

Figura 68

Creación de carpetas

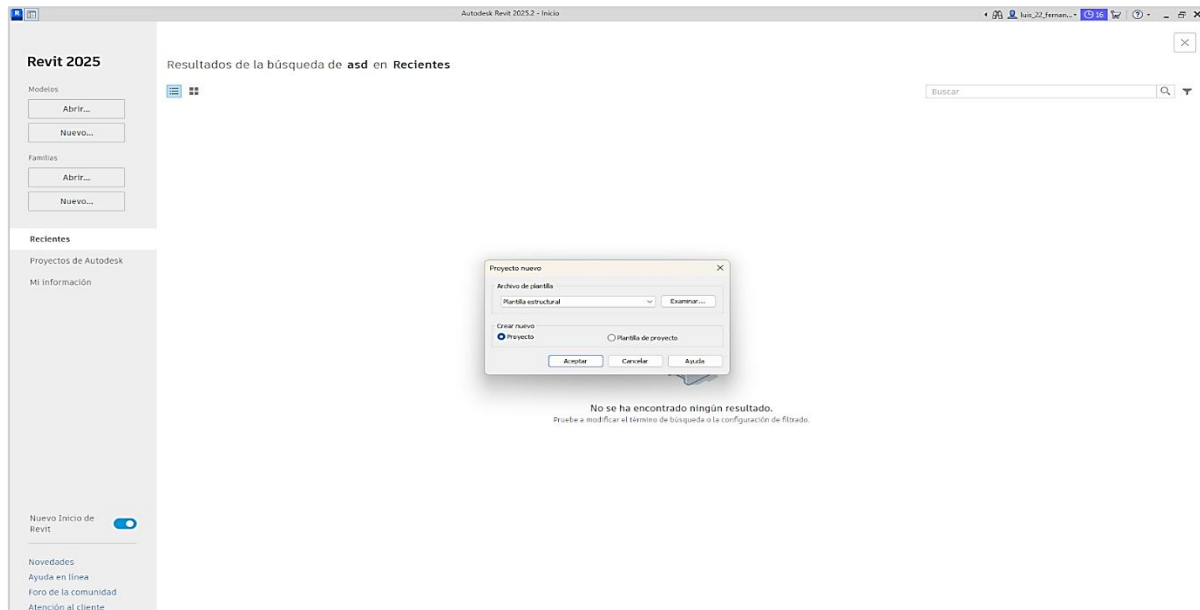


Nota. Elaboración propia.

La figura muestra, la creación de carpetas donde se alojaron los modelos de cada una de las disciplinas, así como de una carpeta donde se alojó los parámetros que permitieron la gestión del mantenimiento de los activos de la I.E.

Figura 69

Inicio del modelo de estructura

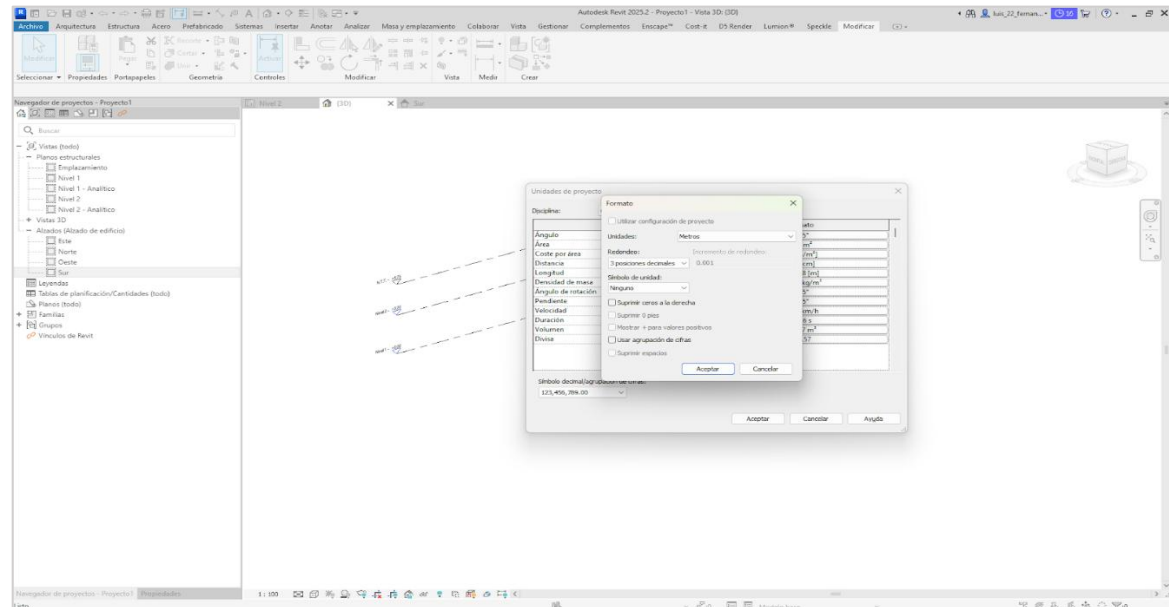


Nota. Datos tomados de Revit 2025

La figura muestra el inicio de la generación del modelo de estructuras, para ello se usó la plantilla por defecto de Revit para estructuras

Figura 70

Configuración previa

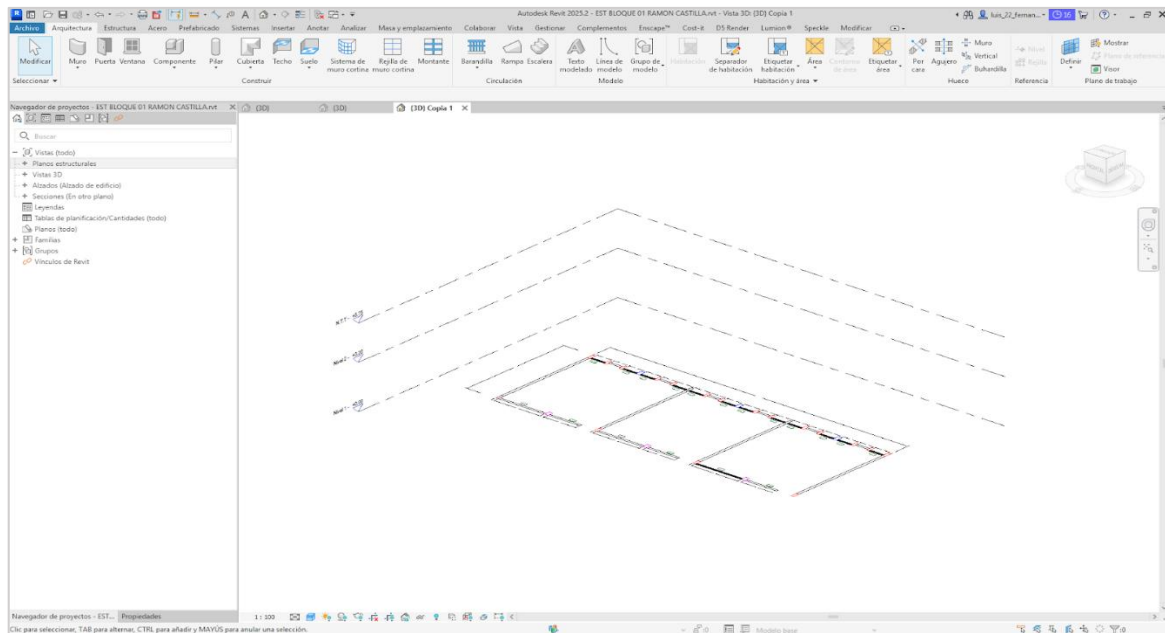


Nota. Datos tomados de Revit 2025.

La figura muestra el inicio con la configuración previa en Revit, niveles, grillas de apoyo, así como la configuración de las unidades.

Figura 71

Modelo en 2D

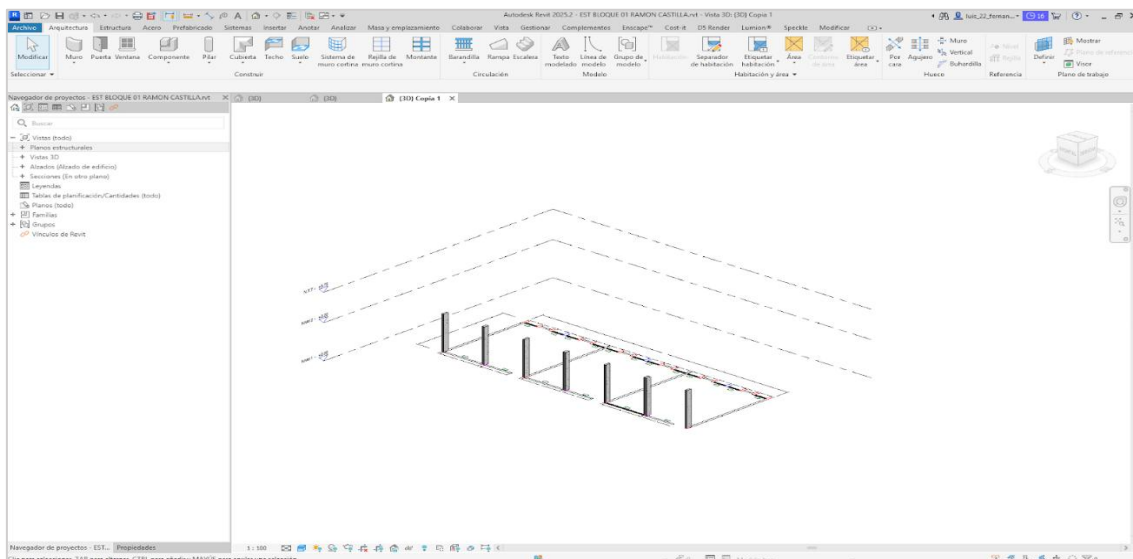


Nota. Datos tomados de Revit 2025.

La figura muestra la vinculación del archivo CAD 2d con los datos exportados de AutoCAD que han sido extraídos en campo.

Figura 72

Modelado de columnas

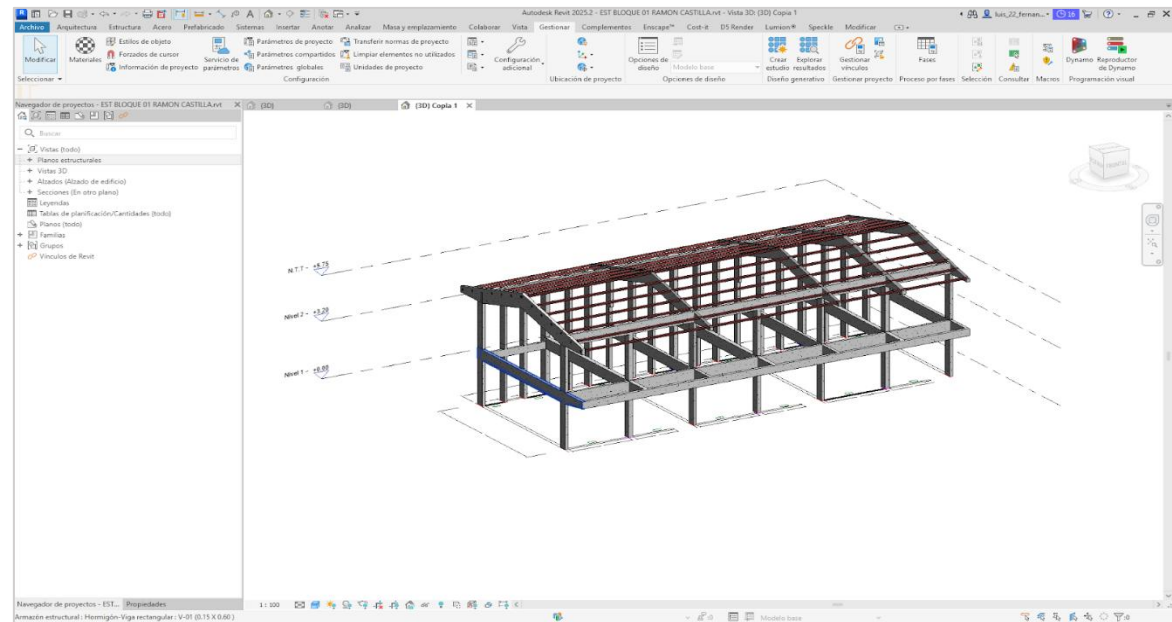


Nota. Datos tomados de Revit 2025.

La figura muestra el modelado de las columnas en 3D, para todos los niveles de la IE.

Figura 73

Modelado de vigas

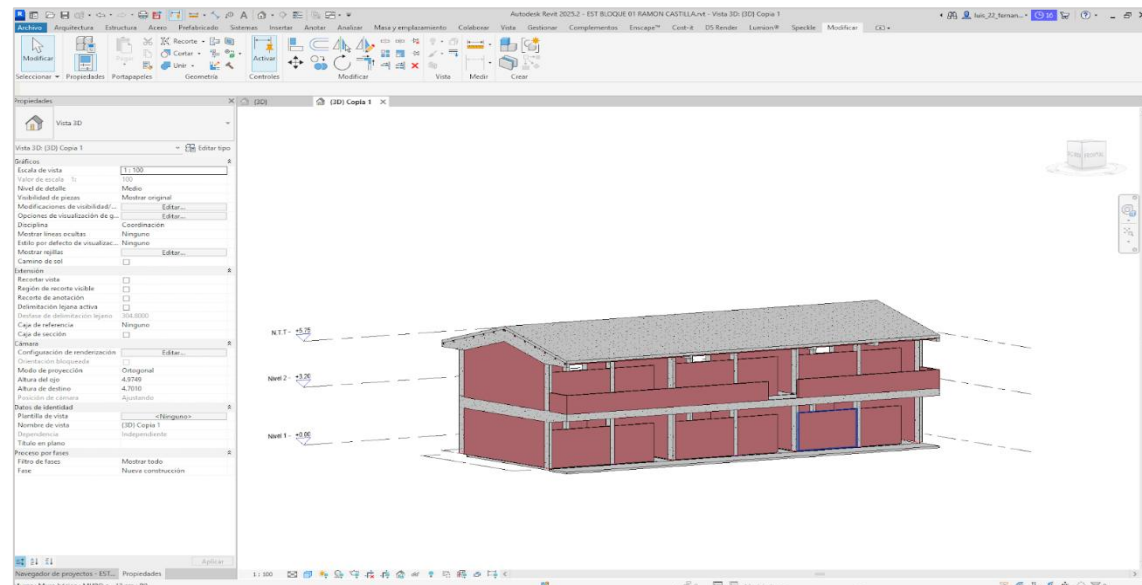


Nota. Datos tomados de Revit 2025.

La figura muestra el modelado de las vigas principales y secundarias en 3D, para todos los niveles de la IE.

Figura 74

Modelado de losa, muros y parapetos

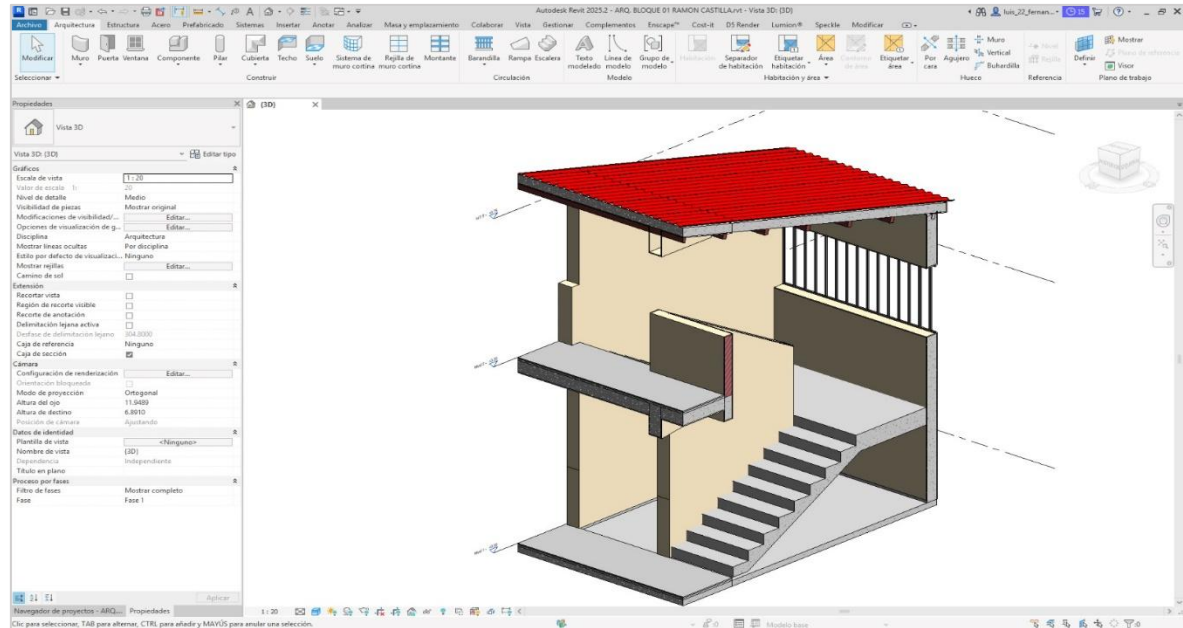


Nota. Datos tomados de Revit 2025.

La figura muestra el modelado de la losa, muros de albañilería y parapetos en 3D, para todos los niveles de la IE.

Figura 75

Modelado de escalera



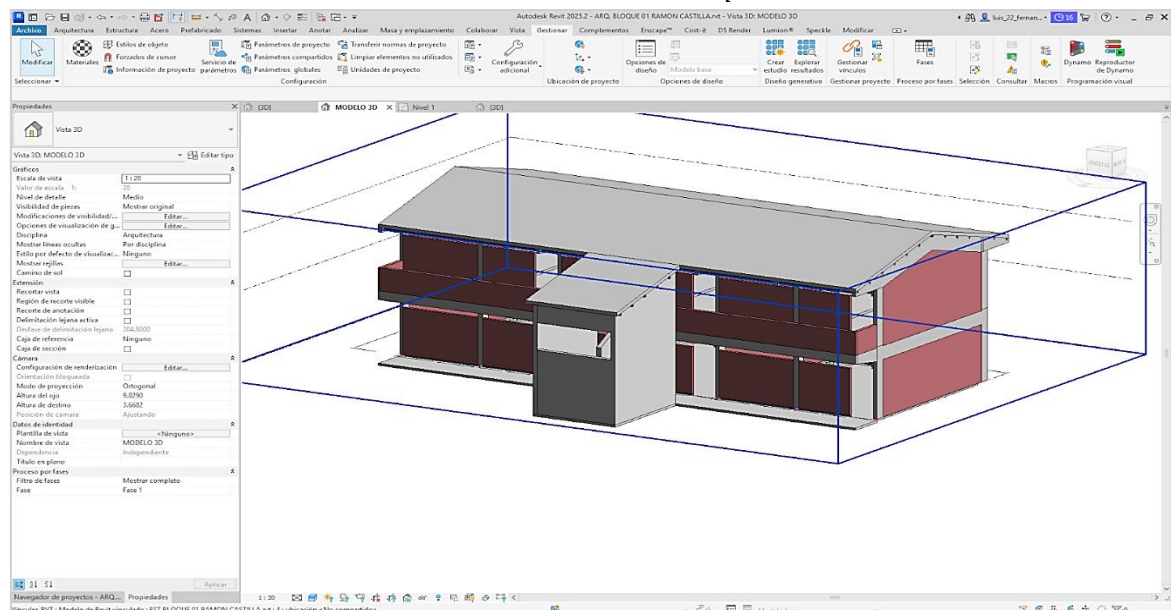
Nota. Datos tomados de Revit 2025.

La figura muestra el modelado de la escalera de dos niveles en 3D, para el acceso al segundo nivel del bloque I.

c. Modelado de arquitectura 3D en software REVIT

Figura 76

Modelo de estructuras vinculado al archivo Revit de arquitectura

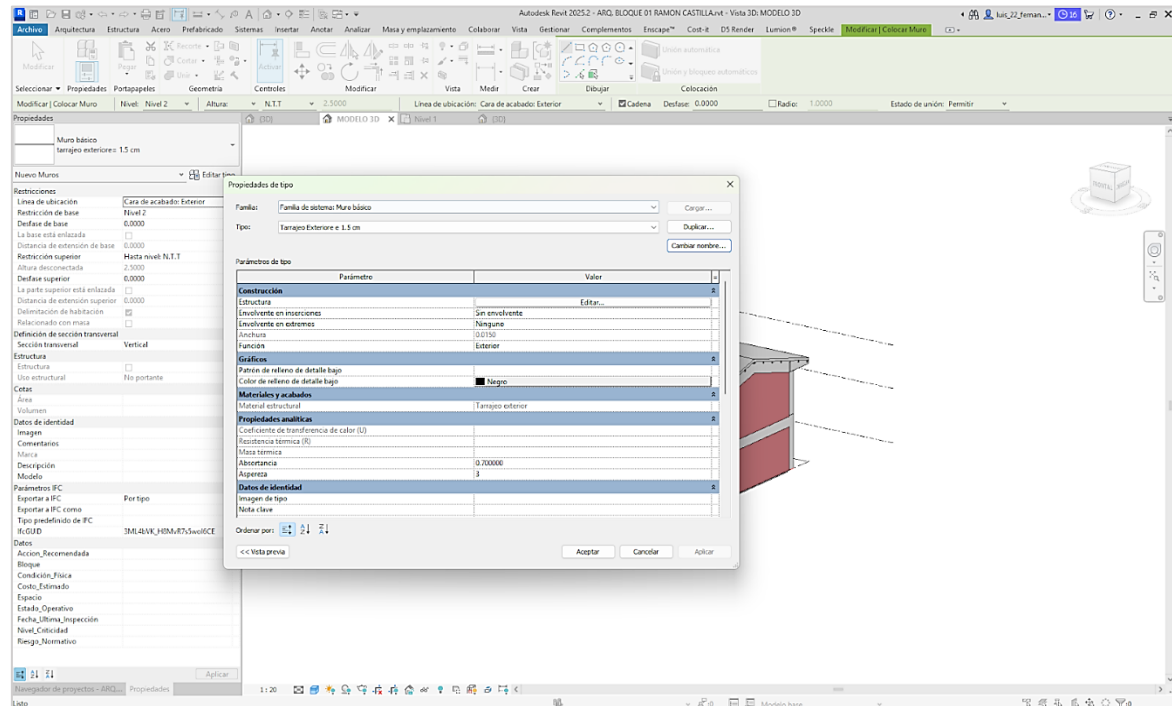


Nota. Datos tomados de Revit 2025.

La figura muestra el modelado de estructuras vinculado al archivo Revit de para el modelado de los elementos arquitectónicos, donde se están consideraron como muros al tarrajeo de un espesor de 1.5 cm, ventanas, puertas, suelos (contrapiso y tarrajeo en losa), pintura y techos, que se gestionaran directamente con el aplicativo.

Figura 77

Codificación del tarrajeo muros

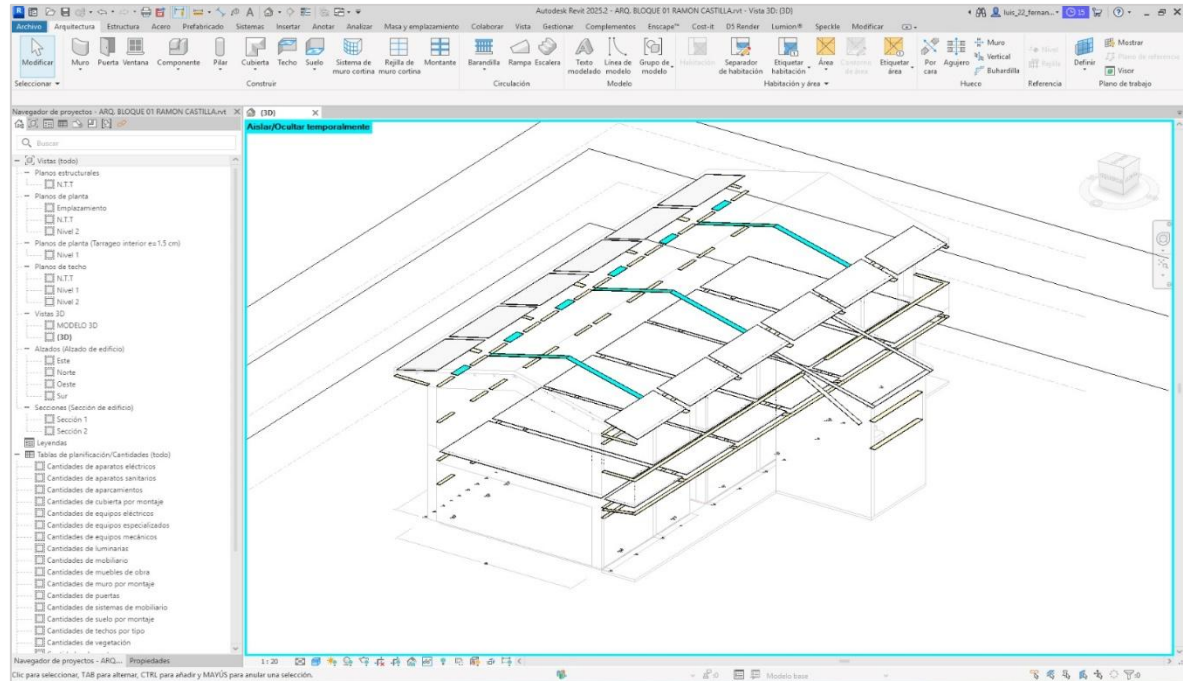


Nota. Datos tomados de Revit 2025.

La figura muestra la creación de los muros que fueron para nuestro modelo el tarrajeo, mismo que estuvo dentro de dos tipos, tanto para tarrajeo interior, como para tarrajeo exterior.

Figura 78

Codificación de suelos (tarrajeo en losas)

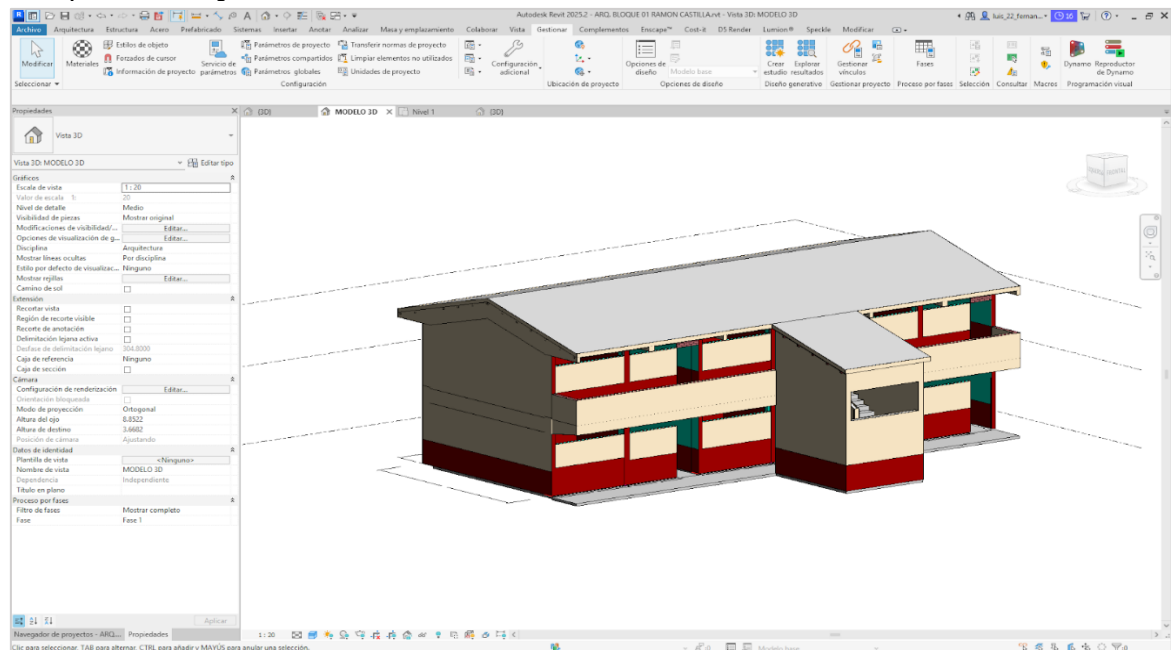


Nota. Datos tomados de Revit 2025.

La figura muestra, la categoría suelos, donde se modeló el tarrajeo en losas y ciellorraso debido al ser elementos de distribución horizontal.

Figura 79

Codificación de pintura

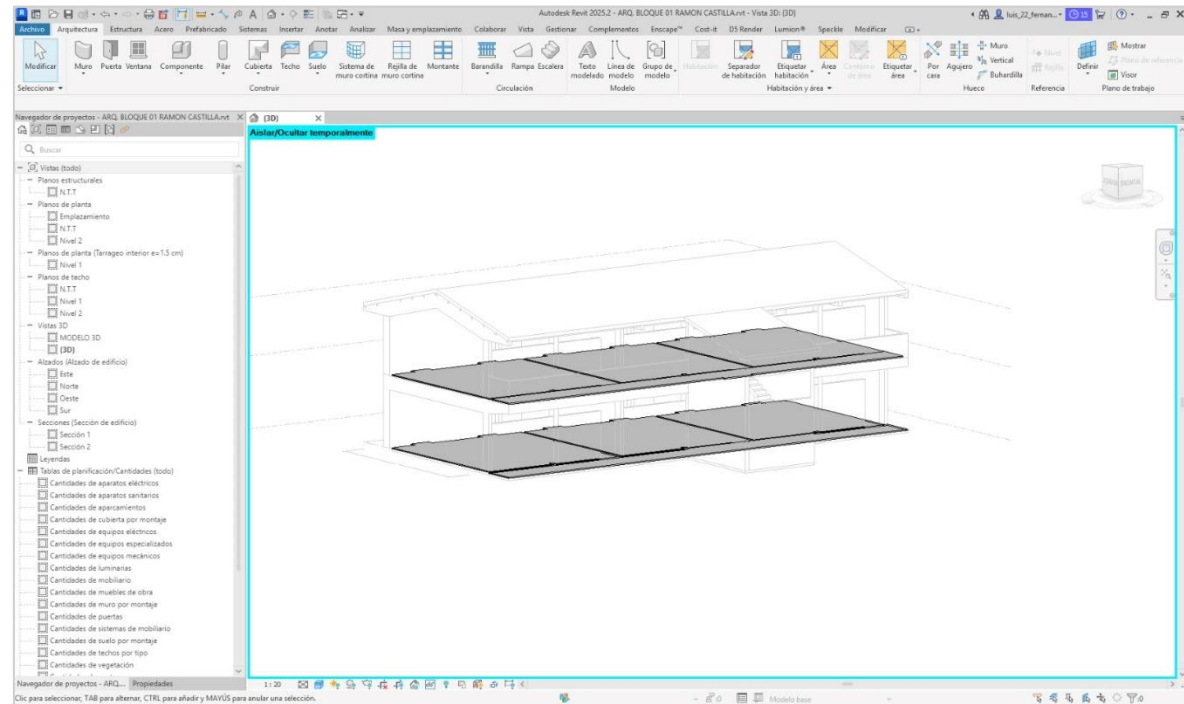


Nota. Datos tomados de Revit 2025.

La figura muestra, la categoría de pintura esmalte, que para no sobre cargar el modelo se utilizó la propiedad de poder colocar un material directamente en el muro (tarrajeo) así cuando se analice el estado del tarrajeo se podrá determinar si el daño es a nivel de pintura o tarrajeo.

Figura 80

Codificación de suelos (contrapisos)

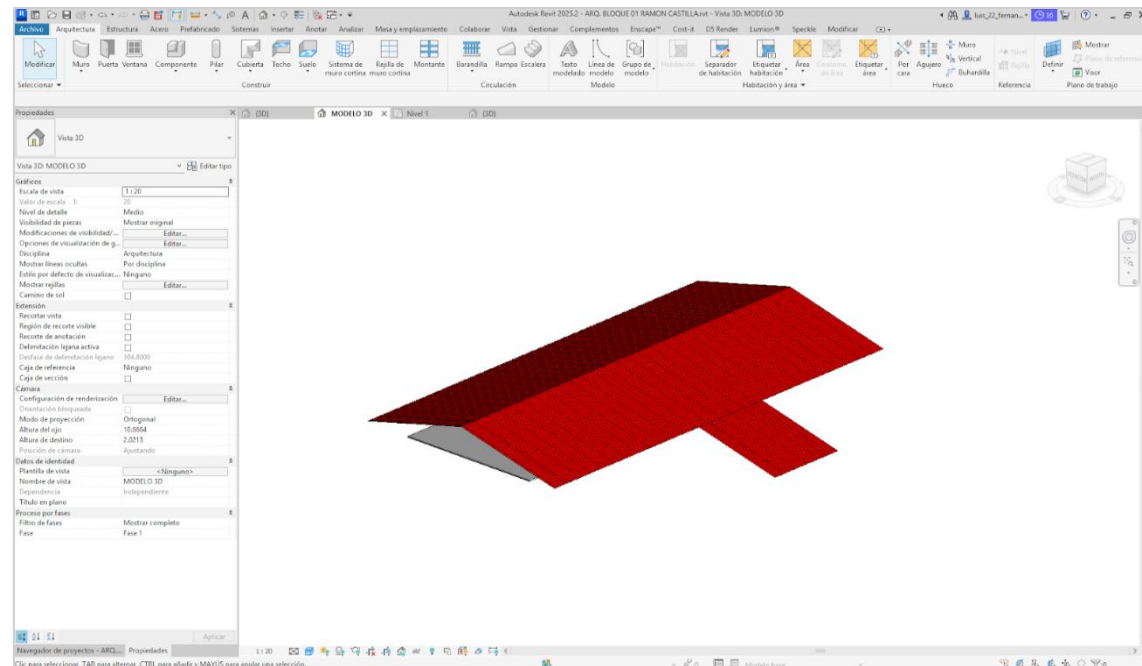


Nota. Datos tomados de Revit 2025.

La figura muestra, la categoría suelos, donde se modeló los contrapisos de 5cm debido al ser elementos de distribución horizontal.

Figura 81

Codificación de cobertura

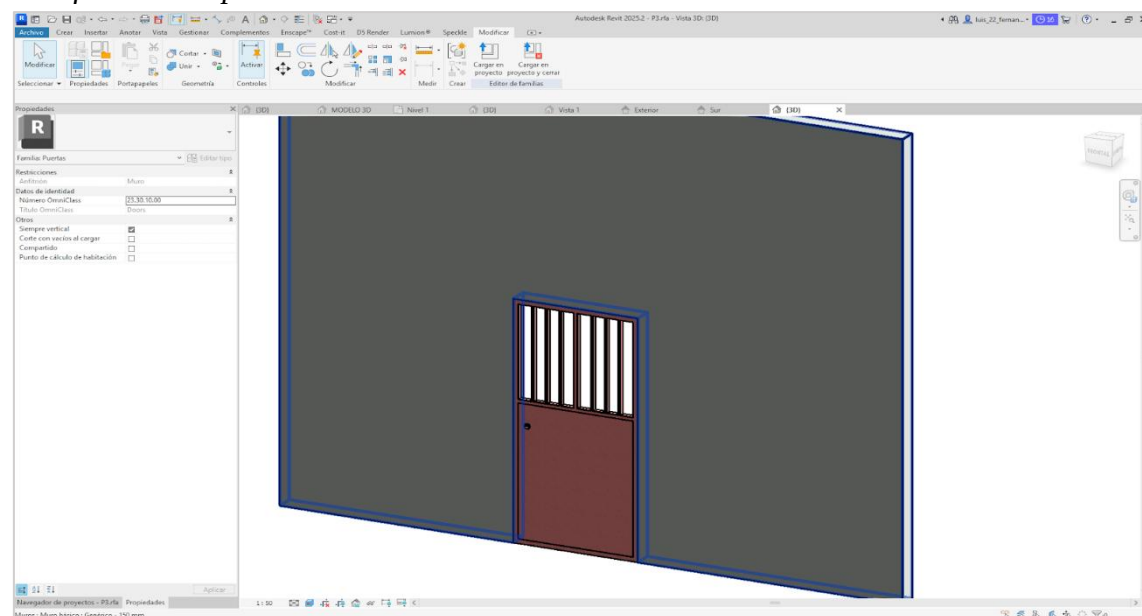


Nota. Datos tomados de Revit 2025.

La figura muestra, la categoría de cobertura tr-4 (Aluzinc) con la que cuenta el bloque y al mismo tiempo se tiene el cielorraso (baldosas de yeso) instalado en las aulas del segundo nivel de la institución.

Figura 82

Codificación de puertas

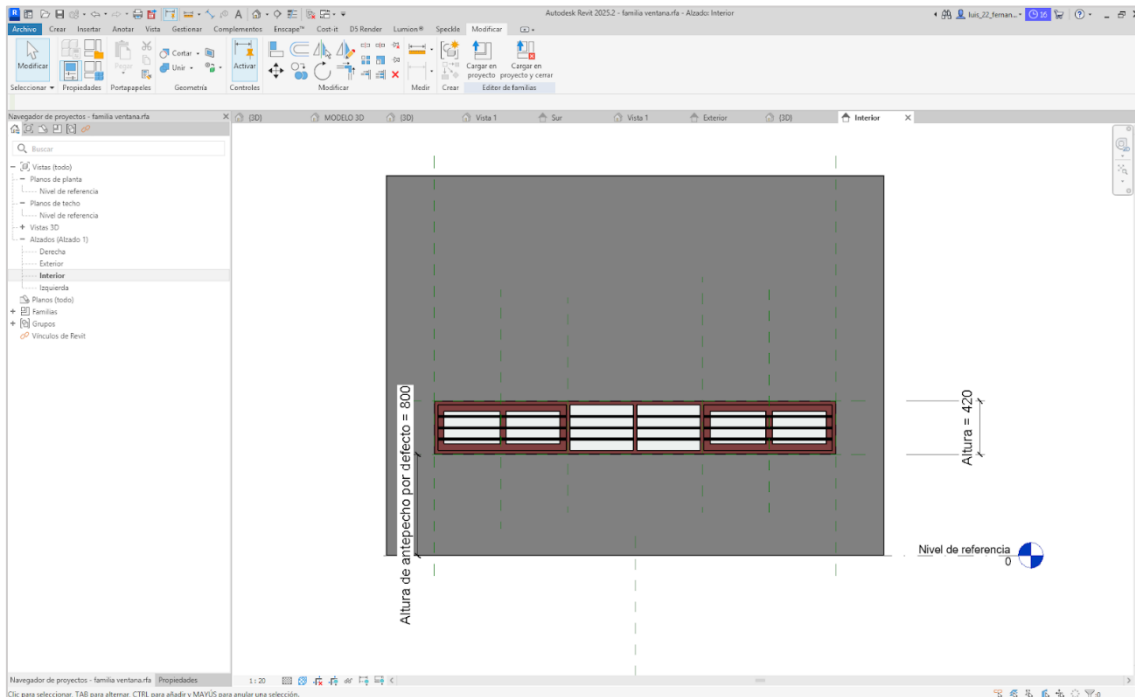


Nota. Datos tomados de Revit 2025.

La figura muestra, la categoría de puertas, donde para nuestro caso solo se contó con un solo tipo de familia que es una puerta metálica de 2.10 m x 1 m, para su creación se usó la plantilla de puerta métrica en Revit donde nos permitió modelar la puerta que posteriormente se insertara en el modelo Revit de arquitectura

Figura 83

Codificación de ventana frontal alta

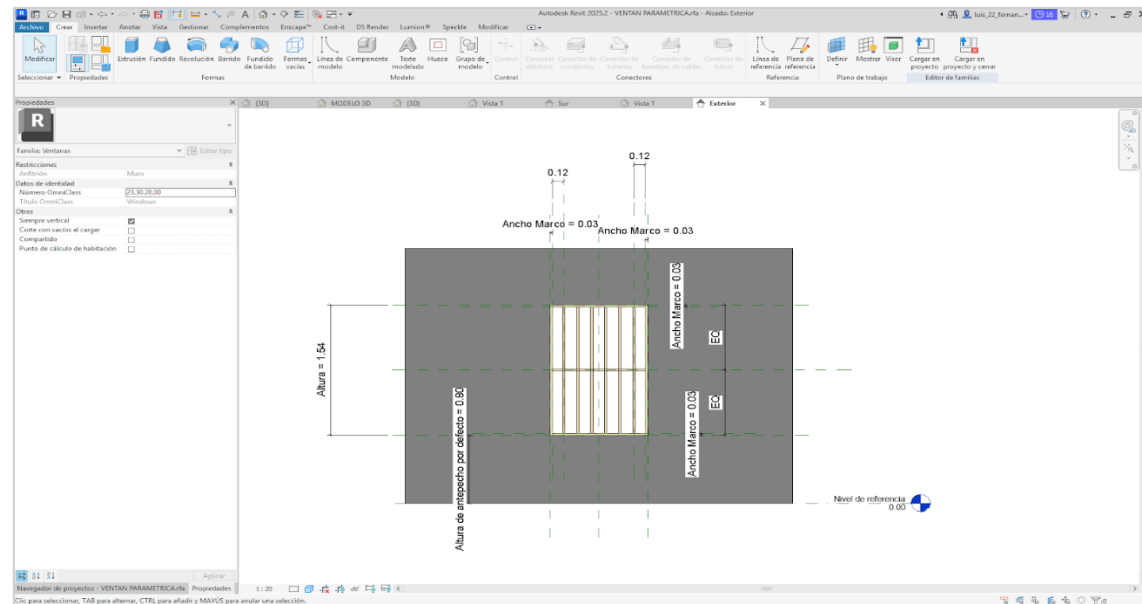


Nota. Datos tomados de Revit 2025.

La figura muestra, la categoría de ventanas, donde la ventana frontal alta fue de 3.23 m x 0.42 m, para su creación se usa la plantilla de ventana métrica en Revit donde nos permitió modelar la ventana que posteriormente se insertara en el modelo Revit.

Figura 84

Codificación de ventana posteriores

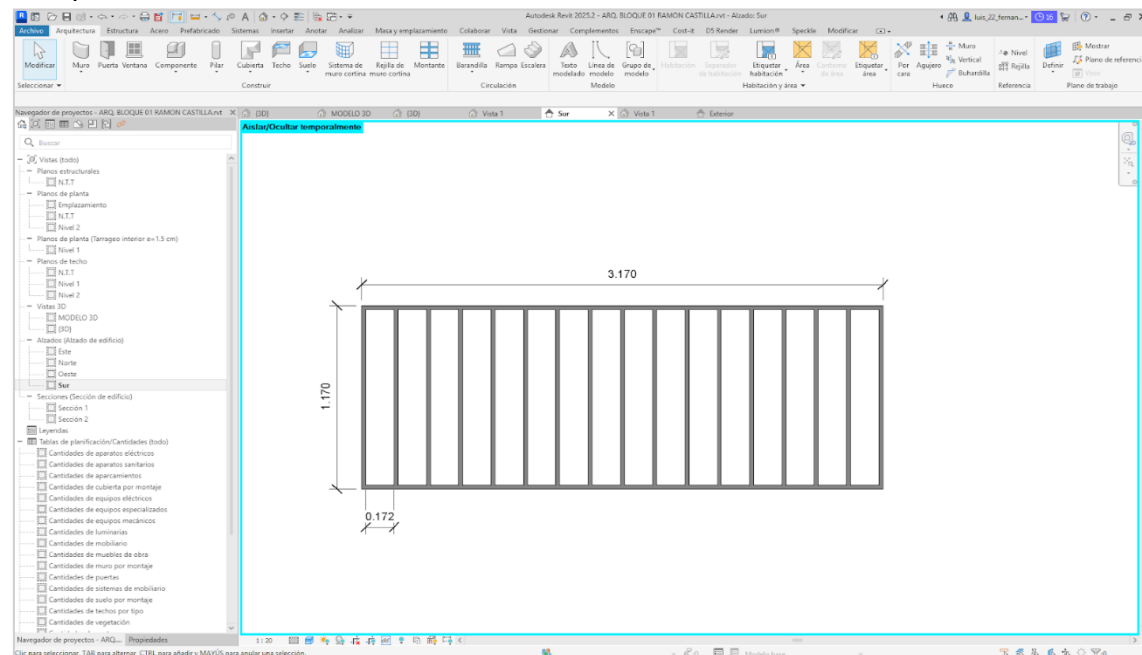


Nota. Datos tomados de Revit 2025.

La figura muestra, la categoría de ventanas, donde la ventana posterior fue de 1.54 m x 0.90m, para su creación se usa la plantilla de ventana métrica en Revit donde nos permitió modelar la ventana que posteriormente se insertara en el modelo Revit.

Figura 85

Codificación de ventana de escalera



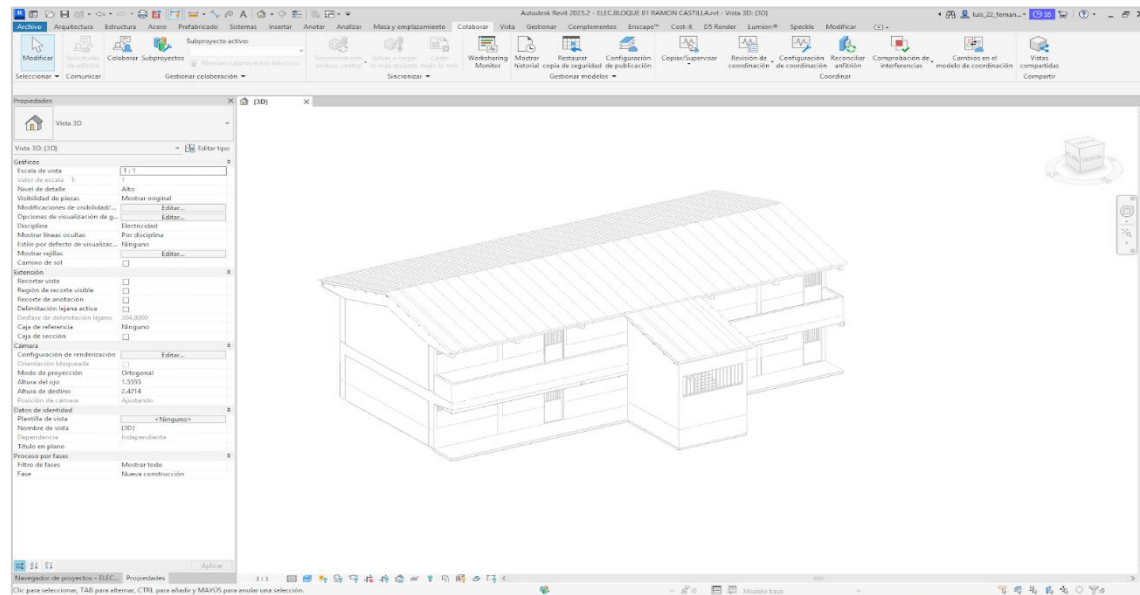
Nota. Datos tomados de Revit 2025.

La figura muestra, la categoría de ventanas, donde la ventana de la escalera fue de 3.17 m x 1.17, para su creación se usa la plantilla de ventana métrica en Revit donde nos permitió modelar la ventana que posteriormente se insertara en el modelo Revit.

d. Modelado de eléctricas 3D en software REVIT

Figura 86

Modelo de estructura vinculado al archivo Revit de estructuras y arquitectura

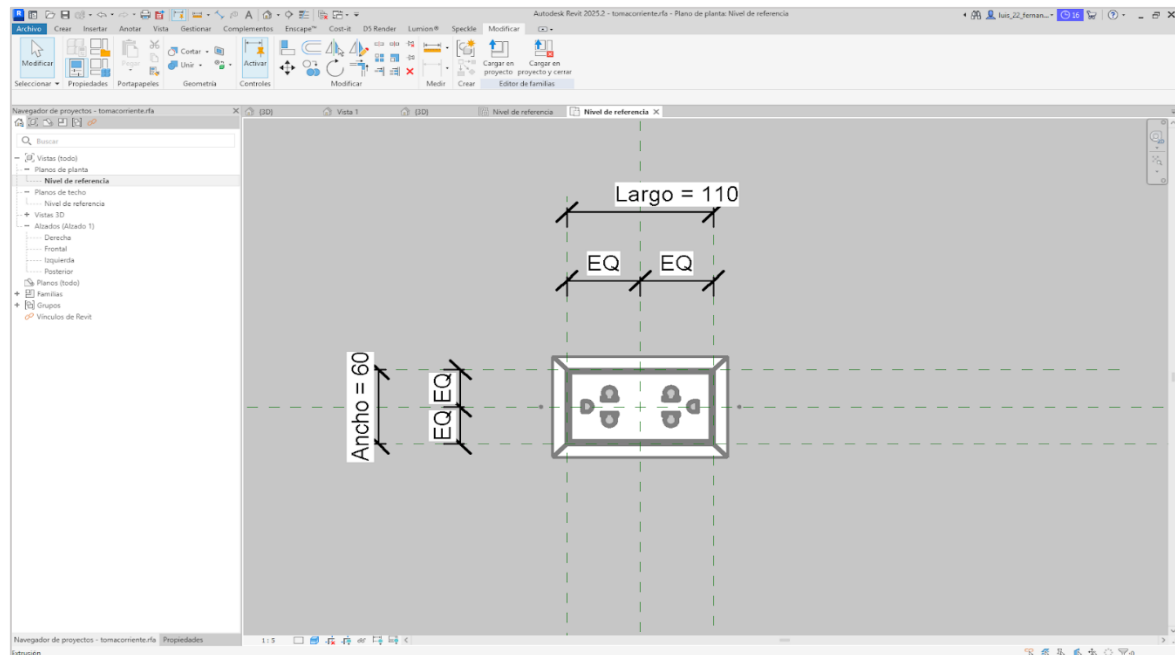


Nota. Datos tomados de Revit 2025.

La figura muestra, la vinculación de las instalaciones eléctricas a la arquitectura y estructuras previamente modelados, mismo que permitió determinar la correcta posición de cada una de los puntos tanto de iluminación como de fuerza. Además, al no encontrarse el registro de los planos de distribución de las tuberías y con fines de referenciación se modeló los conductos a fin de identificar cada punto de la edificación. Se modelaron puntos de potencia, tomacorrientes, para los puntos de iluminación (luminarias led tipo bombillas, paneles de lumina empotradas en el techo y luces empotradas en la pared), los interruptores y finalmente los tableros generales.

Figura 87

Inserción de tomacorriente

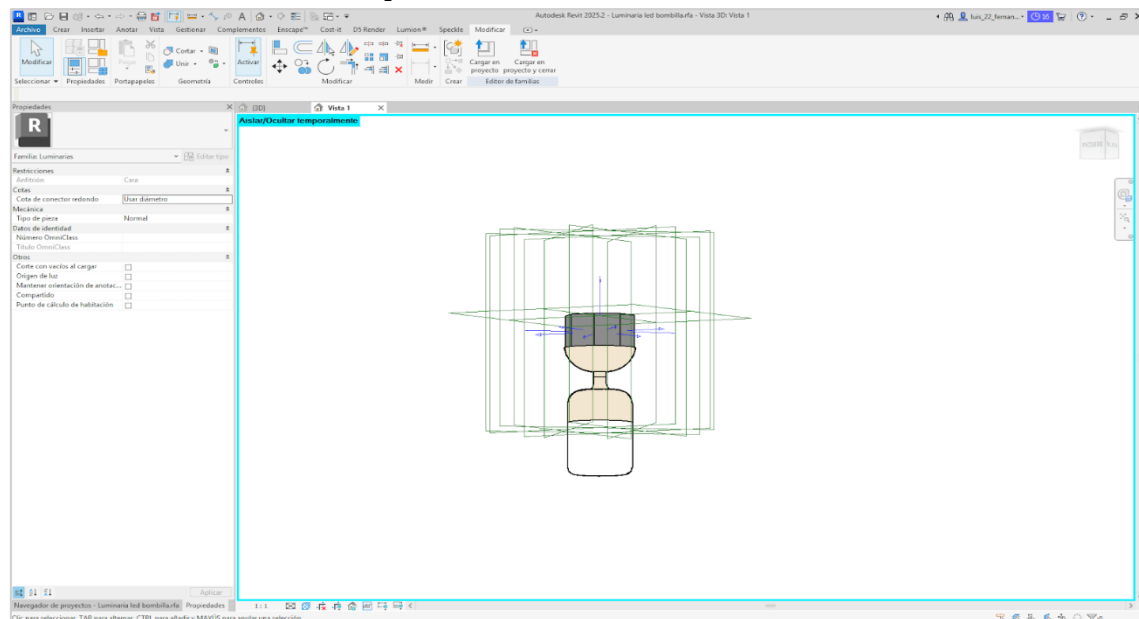


Nota. Datos tomados de Revit 2025.

La figura muestra la inserción de un tomacorriente, con dimensiones paramétricas de 110 mm de largo y 60 mm de ancho, configurado mediante restricciones de igualdad (EQ) que garantizaron su simetría respecto a los ejes de referencia.

Figura 88

Inserción de luminarias led tipo bombillas

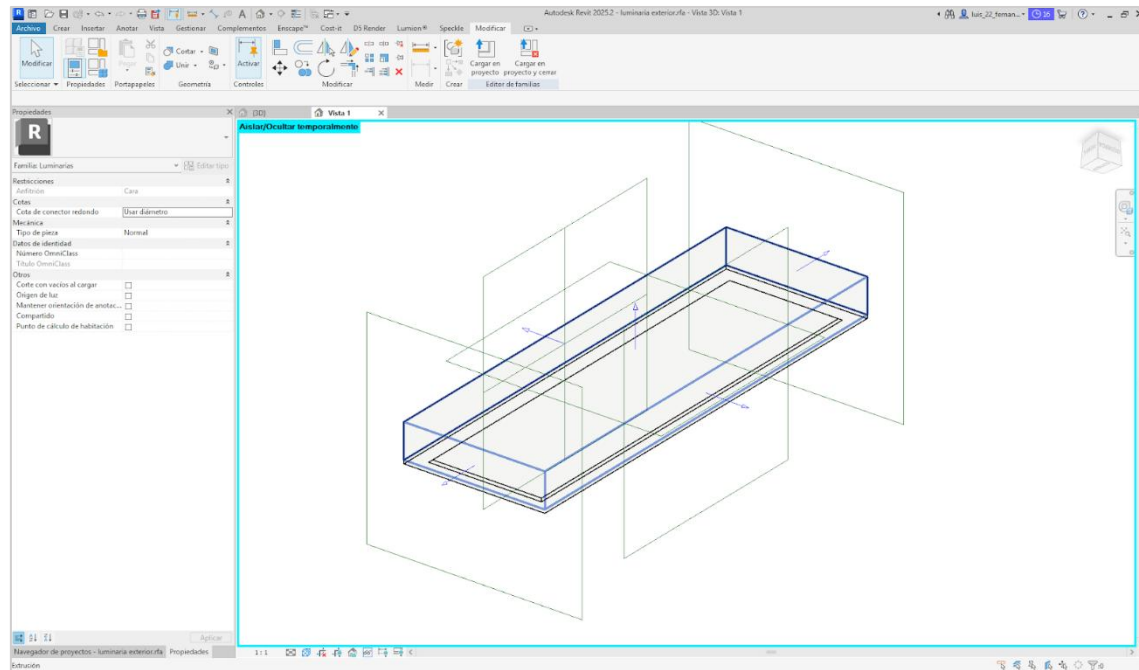


Nota. Datos tomados de Revit 2025.

La figura muestra la inserción de luminarias LED tipo bombilla modeladas, configuradas mediante planos de referencia que permitieron controlar su simetría, altura y ubicación dentro del aula. Este tipo de luminaria proporciona iluminación puntual eficiente y dentro de los ambientes de las aulas.

Figura 89

Inserción de luminarias tipo panel

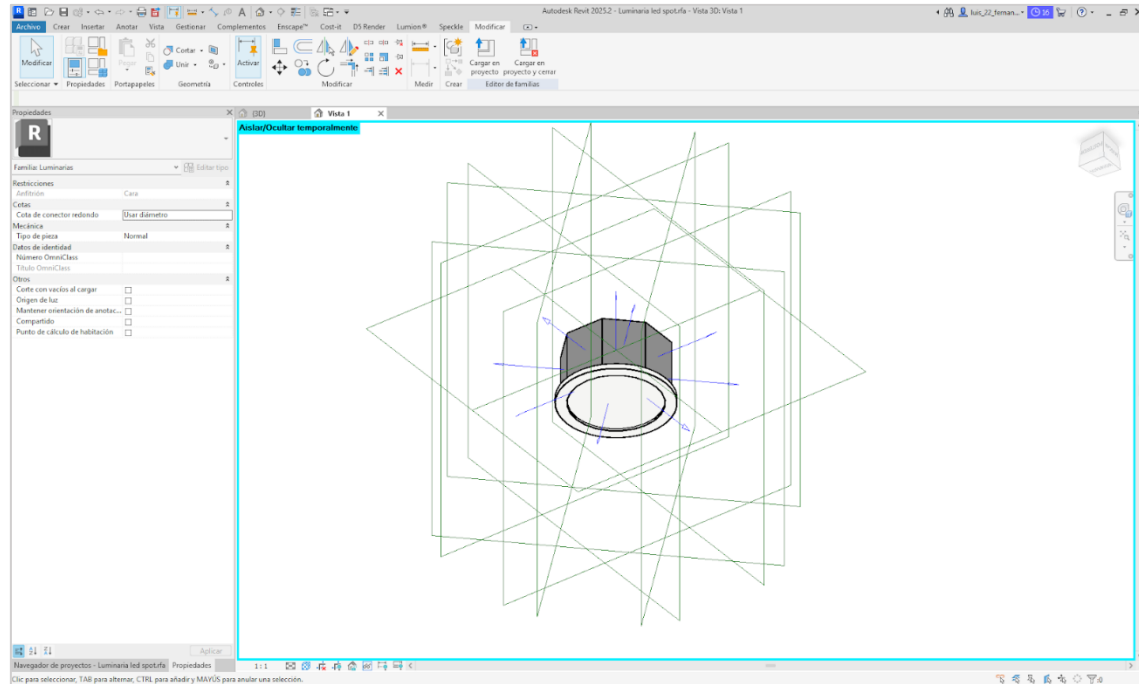


Nota. Datos tomados de Revit 2025.

La figura muestra la inserción de una luminaria tipo panel LED empotrada, para el pasillo del primer nivel. El elemento presentó geometría rectangular alargada y está configurado mediante planos de referencia que permiten su alineación y centrado respecto al eje del ambiente. Este tipo de luminaria garantiza iluminación uniforme, eficiencia energética y adecuada visibilidad en áreas de circulación.

Figura 90

Inserción de luminaria simple circular empotrada

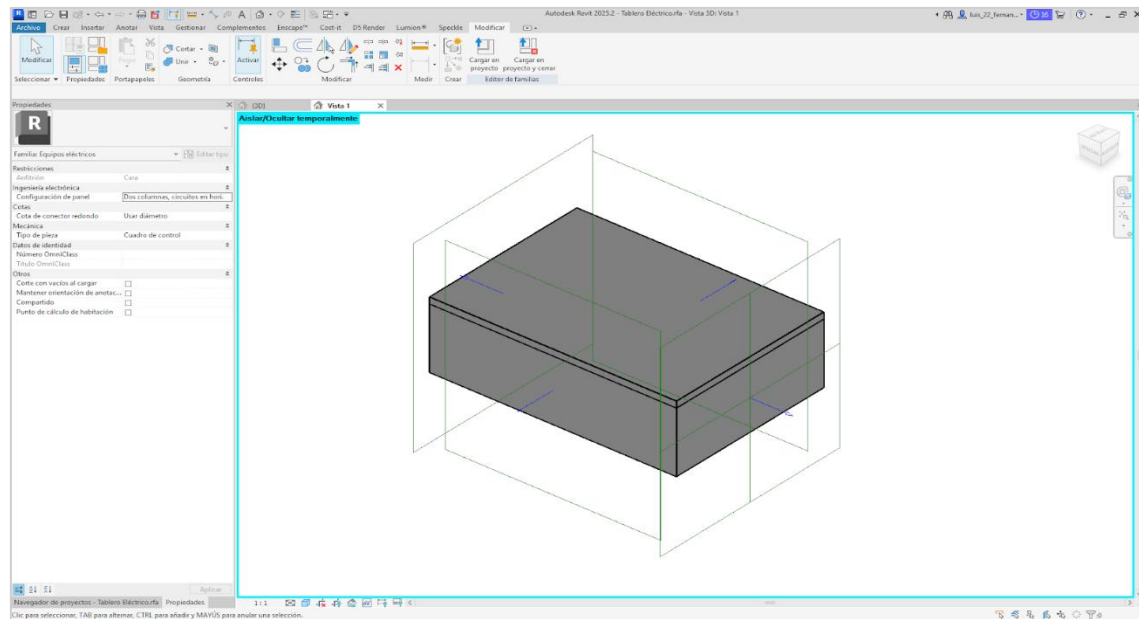


Nota. Datos tomados de Revit 2025.

La figura muestra la inserción de una luminaria simple tipo downlight LED empotrada, para la iluminación del pasillo del segundo nivel y la escalera. El elemento presentó geometría circular y se encuentra configurado mediante planos de referencia que garantizan su simetría, alineación y correcta orientación del flujo luminoso. Este tipo de luminaria proporciona iluminación puntual uniforme, contribuyendo a la seguridad y visibilidad en áreas de circulación vertical y horizontal.

Figura 91

Inserción de tablero eléctrico



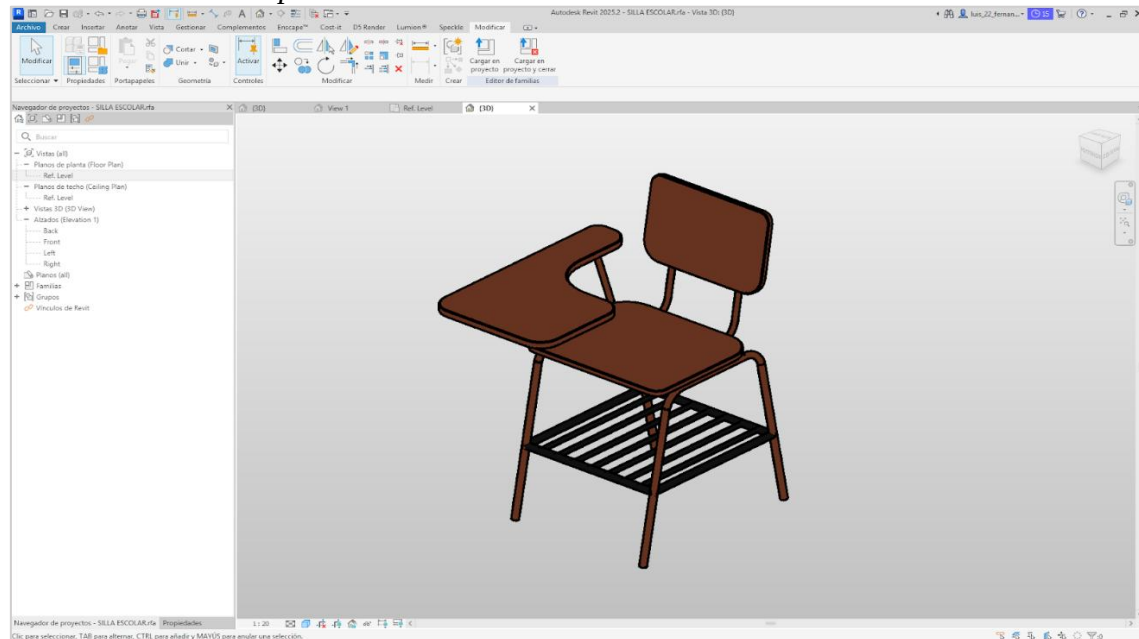
Nota. Datos tomados de Revit 2025.

La figura muestra la inserción del tablero eléctrico empotrado. Garantiza el control general del bloque.

e. Modelado de mobiliario 3D en software REVIT

Figura 92

Inserción de silla unipersonal

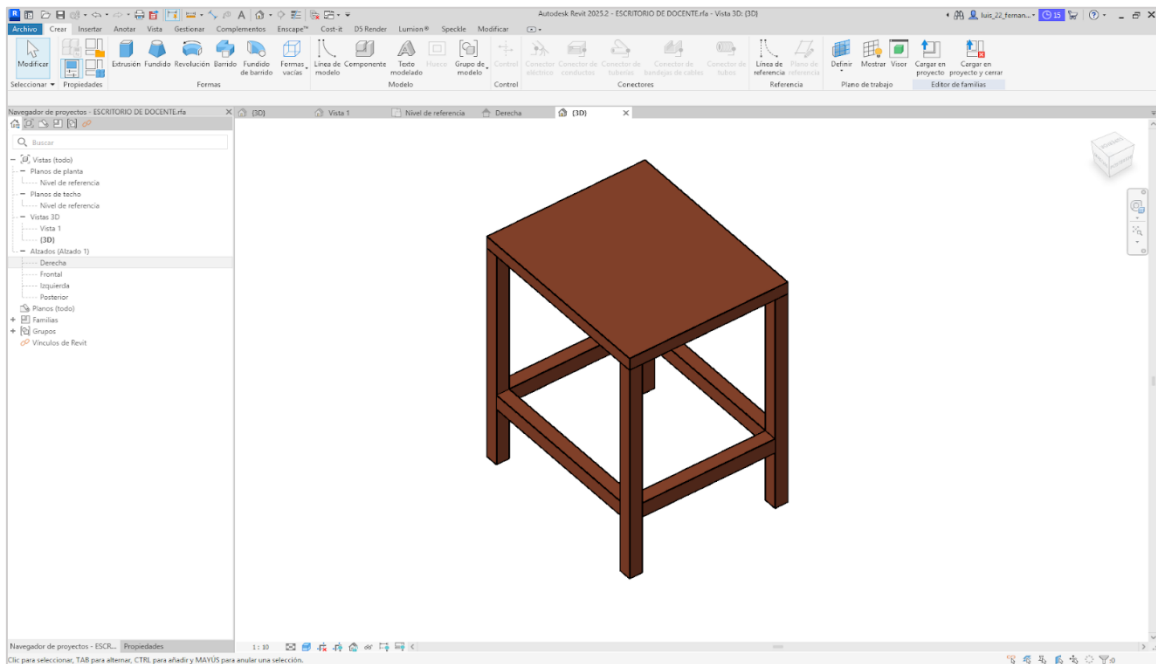


Nota. Datos tomados de Revit 2025.

La figura muestra la inserción de sillas unipersonales para las estudiantes modeladas, configuradas mediante planos de referencia que permitieron controlar su simetría, altura y ubicación dentro del aula.

Figura 93

Inserción de escritorio docente

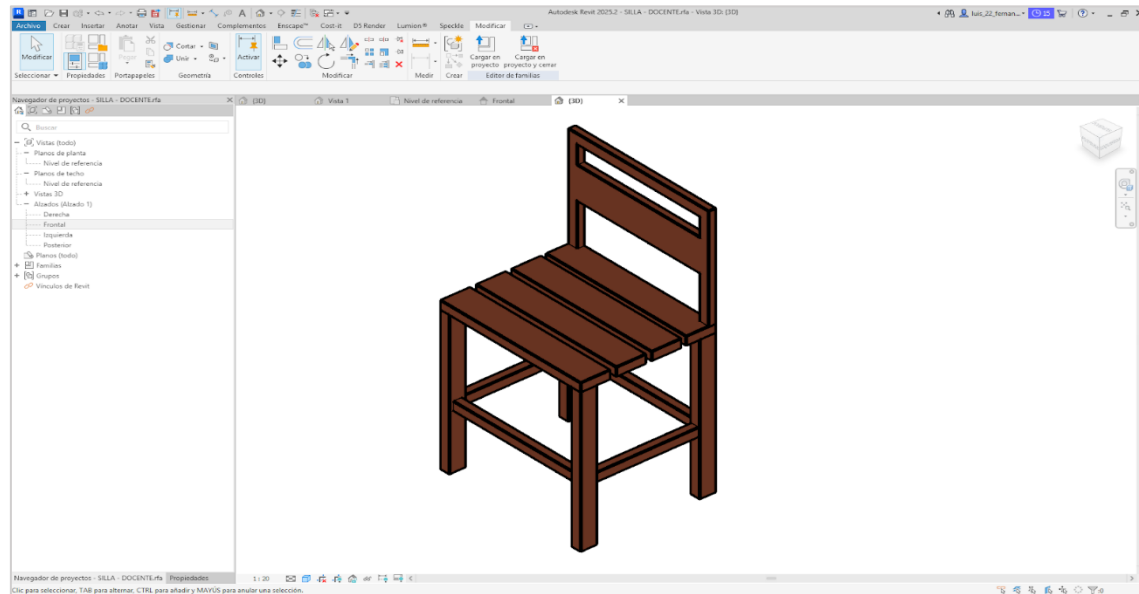


Nota. Datos tomados de Revit 2025.

La figura muestra la inserción de escritorio docente modeladas, configuradas mediante planos de referencia que permitieron controlar su simetría, altura y ubicación dentro del aula.

Figura 94

Inserción de silla docente

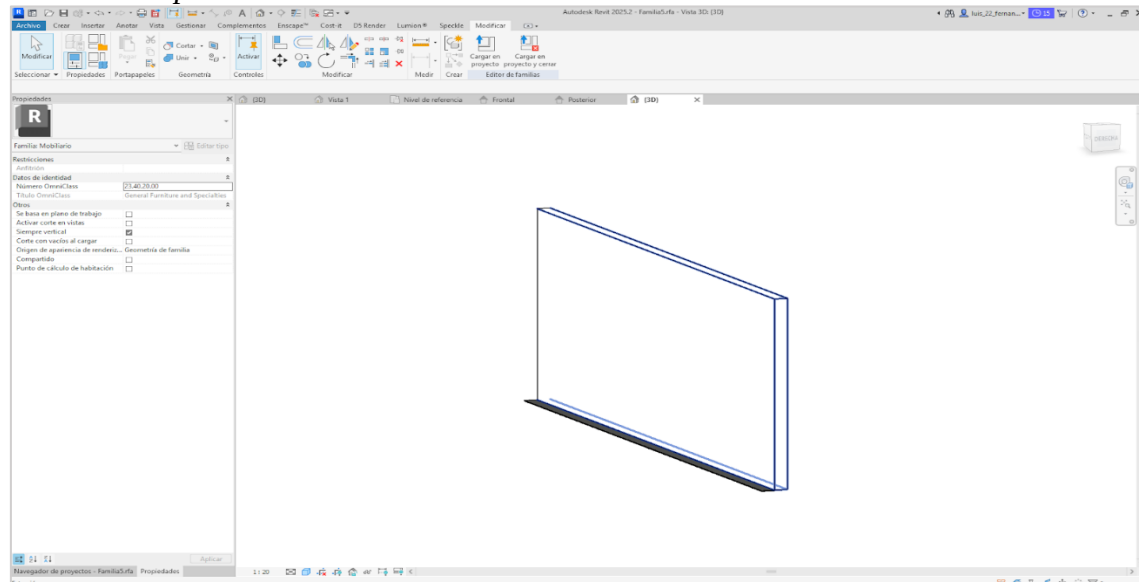


Nota. Datos tomados de Revit 2025.

La figura muestra la inserción de silla docente modeladas, configuradas mediante planos de referencia que permitieron controlar su simetría, altura y ubicación dentro del aula.

Figura 95

Inserción de pizarra acrílica

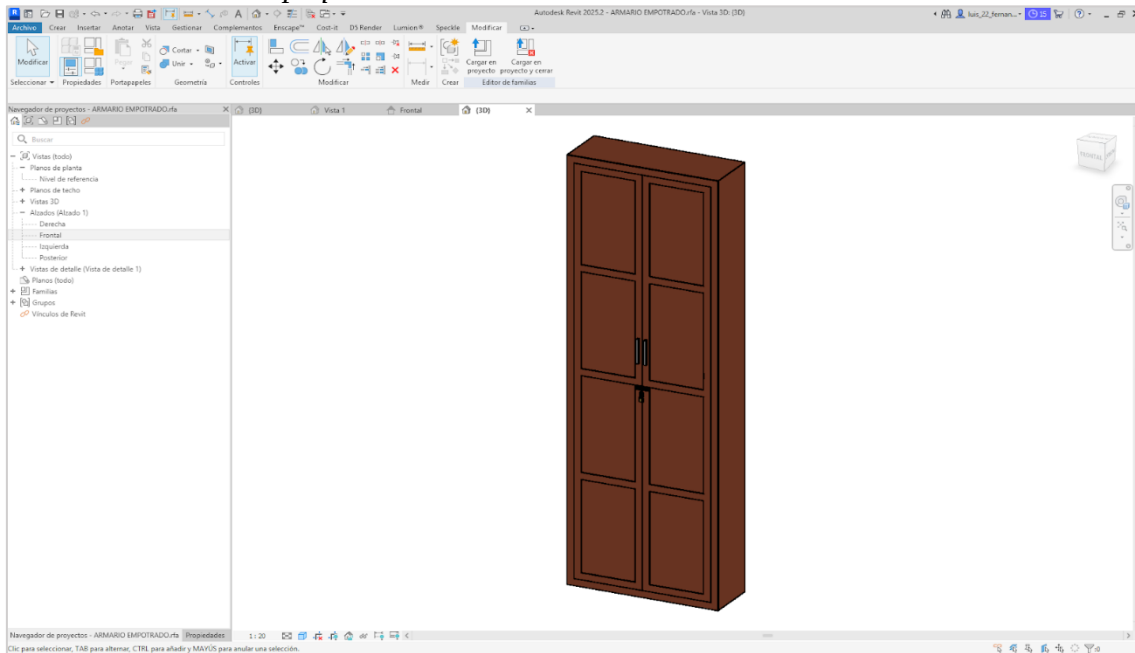


Nota. Datos tomados de Revit 2025.

La figura muestra la inserción de pizarra acrílica modeladas, configuradas mediante planos de referencia que permitieron controlar su simetría, altura y ubicación dentro del aula.

Figura 96

Inserción de armario pequeño

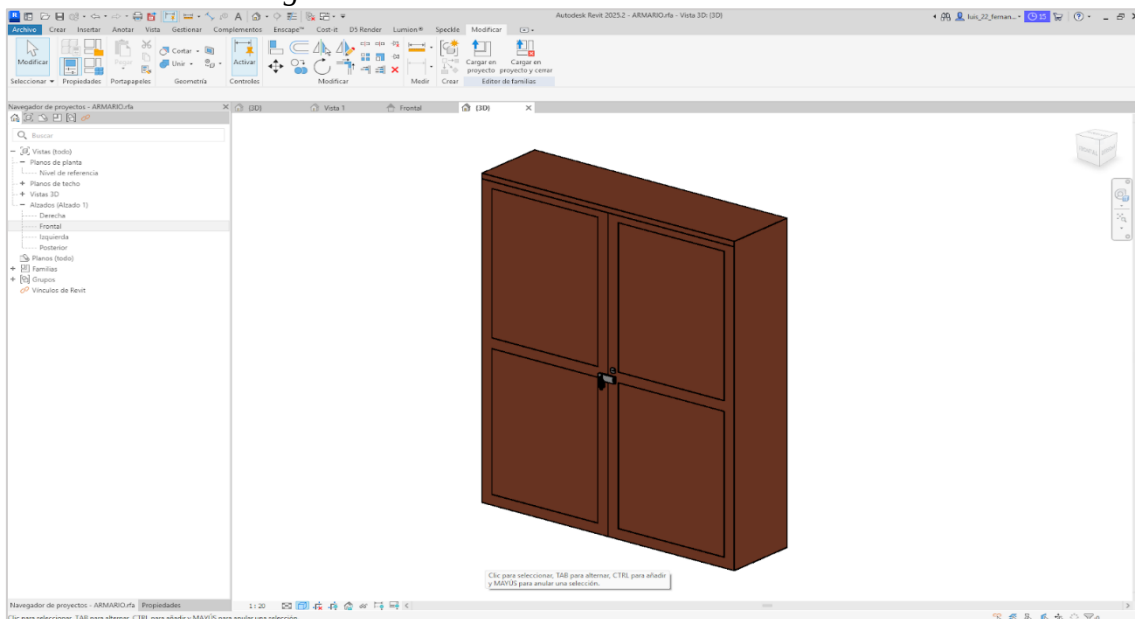


Nota. Datos tomados de Revit 2025.

La figura muestra la inserción de armario de madera de 2.45 x 0.85 x 0.30m, configuradas mediante planos de referencia que permitieron controlar su simetría, altura y ubicación dentro del aula.

Figura 97

Inserción de armario grande

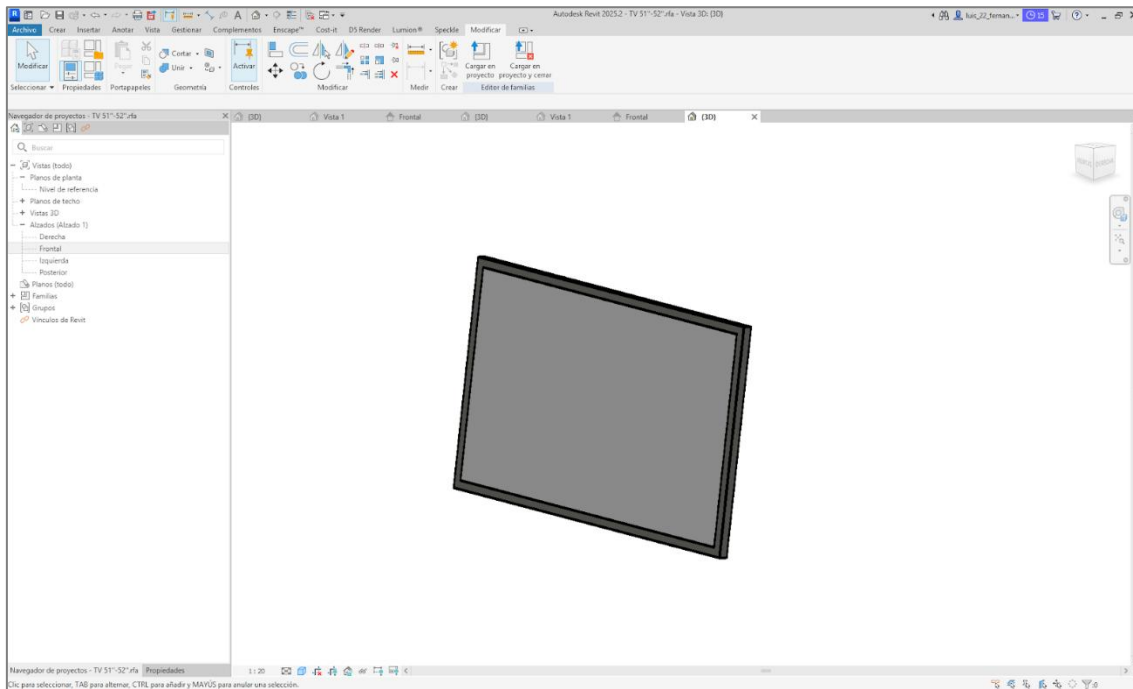


Nota. Datos tomados de Revit 2025.

La figura muestra la inserción de armario de madera 1.85 x 1.50 x 0.40m, configuradas mediante planos de referencia que permitieron controlar su simetría, altura y ubicación dentro del aula

Figura 98

Inserción de TV



Nota. Datos tomados de Revit 2025.

La figura muestra la inserción de TV de madera de 1.13 x 0.65m, configuradas mediante planos de referencia que permiten controlar su simetría, altura y ubicación dentro del aula.

2.3.4.4. FASE 4: Integración del Revit con el Modelo BIM

En esta fase se desarrolló el proceso de interoperabilidad entre el modelo BIM elaborado en Autodesk Revit y el aplicativo diseñado para la gestión del mantenimiento de la Institución Educativa Ramón Castilla y Marquesado. El objetivo fue estructurar, extraer, organizar e integrar información técnica de los activos modelados, permitiendo una gestión basada en datos cuantificables.

a. Definición de parámetros para la gestión del mantenimiento

Con los activos previamente modelados en las disciplinas de arquitectura, instalaciones eléctricas y mobiliario, se procedió a la creación de parámetros que permitan transferir información entre Revit y el aplicativo. Estos parámetros fueron clasificados en cinco grupos:

Parámetros de Identificación del Activo

- ID_Activo
- Disciplina
- Categoria_Activo
- Tipo_Activo
- Marca
- Modelo
- Bloque
- Nivel
- Espacio

Estos campos permiten la trazabilidad técnica dentro del modelo BIM y evitan duplicidades en la base de datos.

Parámetros de Evaluación Técnica

- Condicion_Fisica
- Estado_Operativo

Permiten evaluar el nivel de deterioro observable y la capacidad funcional del activo.

Parámetro de Riesgo Normativo

- Riesgo_Normativo

Incorpora la dimensión de cumplimiento técnico y de seguridad dentro del análisis.

Parámetro Sintetizador

- Nivel_Criticidad

Integra múltiples variables en un indicador cuantitativo para priorización.

Parámetros Económicos y Temporales

- Costo_Estimado
- Fecha_Ultima_Inspeccion
- Fecha_Proxima_Inspeccion
- Frecuencia_Meses

Permiten incorporar planificación presupuestal y mantenimiento preventivo.

b. Modelo Matemático Aplicado para el nivel de criticidad

El nivel de criticidad se determina mediante la siguiente expresión ponderada:

$$\text{Criticidad} = (\text{Condición Física} \times 0.5) + (\text{Riesgo Normativo} \times 0.3) + (\text{Estado Operativo} \times 0.2)$$

$$\text{Criticidad} = (\text{Condición Física} \times 0.5) + (\text{Riesgo Normativo} \times 0.3) + (\text{Estado Operativo} \times 0.2)$$

$$\text{Criticidad} = (\text{Condición Física} \times 0.5) + (\text{Riesgo Normativo} \times 0.3) + (\text{Estado Operativo} \times 0.2)$$

Donde cada variable cualitativa es transformada a valores numéricos según las siguientes tablas.

Tabla 1
Conversión de Condición Física

Condición Física	Valor Numérico	Criterio Técnico
Bueno	1	El activo presenta funcionamiento adecuado y sin deterioro significativo.
Regular	3	Presenta desgaste visible que puede afectar su desempeño a mediano plazo.
Malo	5	Presenta deterioro crítico que compromete su funcionamiento o seguridad.

Tabla 2
Conversión de Riesgo Normativo

Riesgo Normativo	Valor Numérico	Criterio Técnico
Bajo	1	No representa incumplimiento relevante de normativa educativa o de seguridad.
Medio	3	Existe posible incumplimiento parcial o condición mejorable.
Alto	5	Existe incumplimiento que afecta seguridad, salubridad o normativa vigente.

Tabla 3
Conversión de Estado Operativo

Estado Operativo	Valor Numérico	Criterio Técnico
Operativo	1	El activo cumple su función dentro del entorno educativo.
Fuera de servicio	5	El activo no cumple su función, afectando la operatividad del colegio.

Tabla 4
Clasificación Final del Nivel de Criticidad

Resultado Numérico	Nivel de Criticidad	Interpretación Técnica
1.0 – 2.0	Baja	No requiere intervención inmediata. Puede programarse dentro del plan anual de mantenimiento.
2.1 – 3.5	Media	Requiere atención prioritaria en el corto plazo.
3.6 – 5.0	Alta	Requiere intervención urgente debido a riesgo técnico o normativo significativo.

c. Justificación Técnica para la Selección de Parámetros

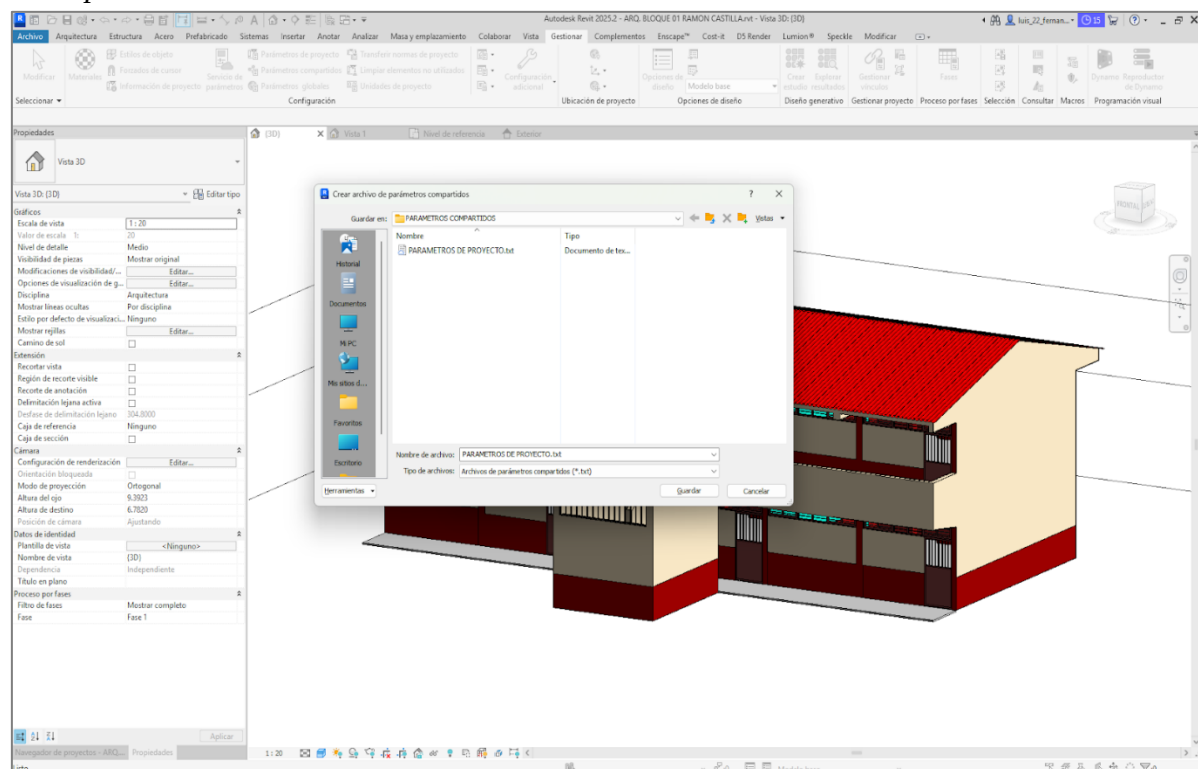
La selección de parámetros fue realizada bajo criterios de relevancia técnica, trazabilidad y aplicabilidad dentro de un entorno BIM. Se priorizaron variables que permiten evaluar el estado físico, operatividad y cumplimiento normativo de los activos de infraestructura educativa, garantizando un modelo cuantificable de priorización de mantenimiento. Esta estructura posibilita la integración con herramientas de análisis de datos y la toma de decisiones basada en información objetiva.

e. Creación de parámetros en revit

Para crear los parámetros ya definidos dentro de Revit, se usó los parámetros compartidos a fin de que se puedan usar en los tres archivos de las tres disciplinas a trabajar, arquitectura, eléctricas y mobiliario.

Figura 99

Creación del archivo de parámetros compartidos (.txt) en Autodesk Revit para interoperabilidad BIM.

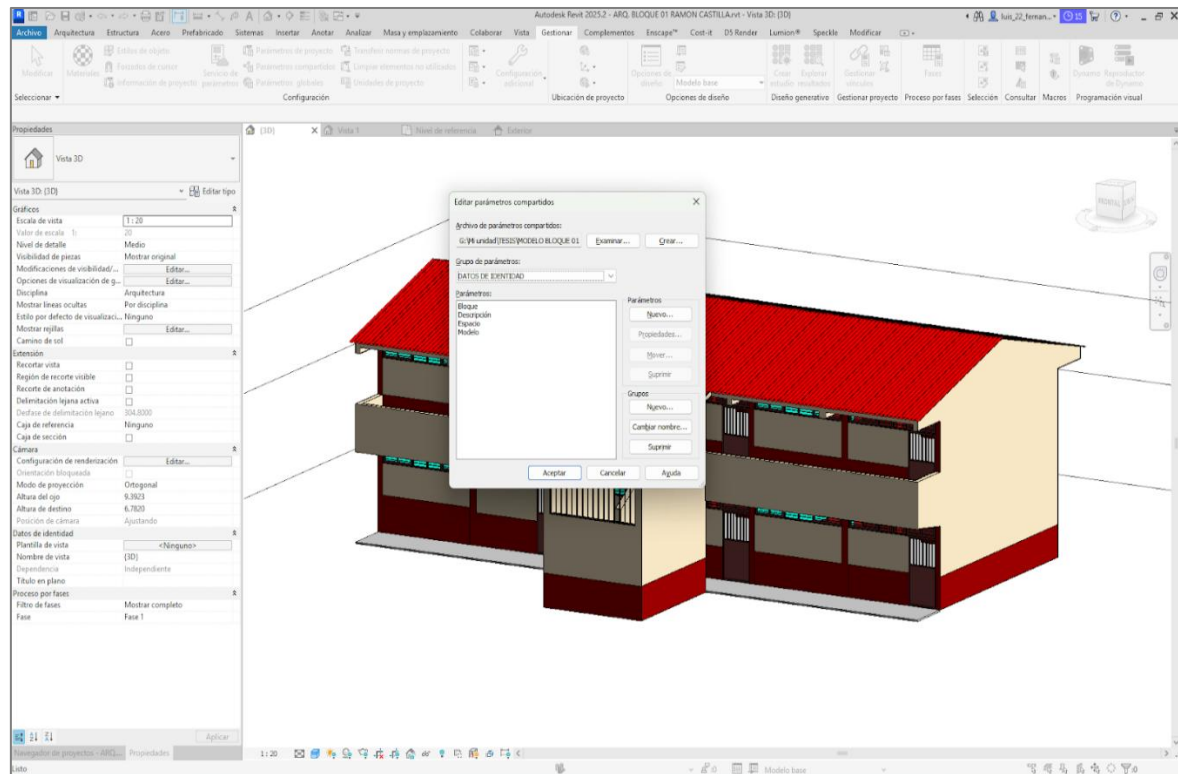


Nota. Datos tomados de Revit 2025.

La figura muestra la creación del archivo de parámetros compartidos en formato .txt, el cual almacena la estructura de datos utilizada para la gestión del mantenimiento. Este archivo permite que los parámetros definidos sean compartidos entre los modelos de arquitectura, instalaciones eléctricas y mobiliario, garantizando interoperabilidad, estandarización y coherencia de la información dentro del entorno BIM. La implementación de parámetros compartidos constituyó la base técnica para la posterior extracción automatizada de datos mediante Dynamo y su integración con el aplicativo desarrollado.

Figura 100

Organización de parámetros compartidos en grupos “Datos de Identidad” y “Gestión del Mantenimiento”.

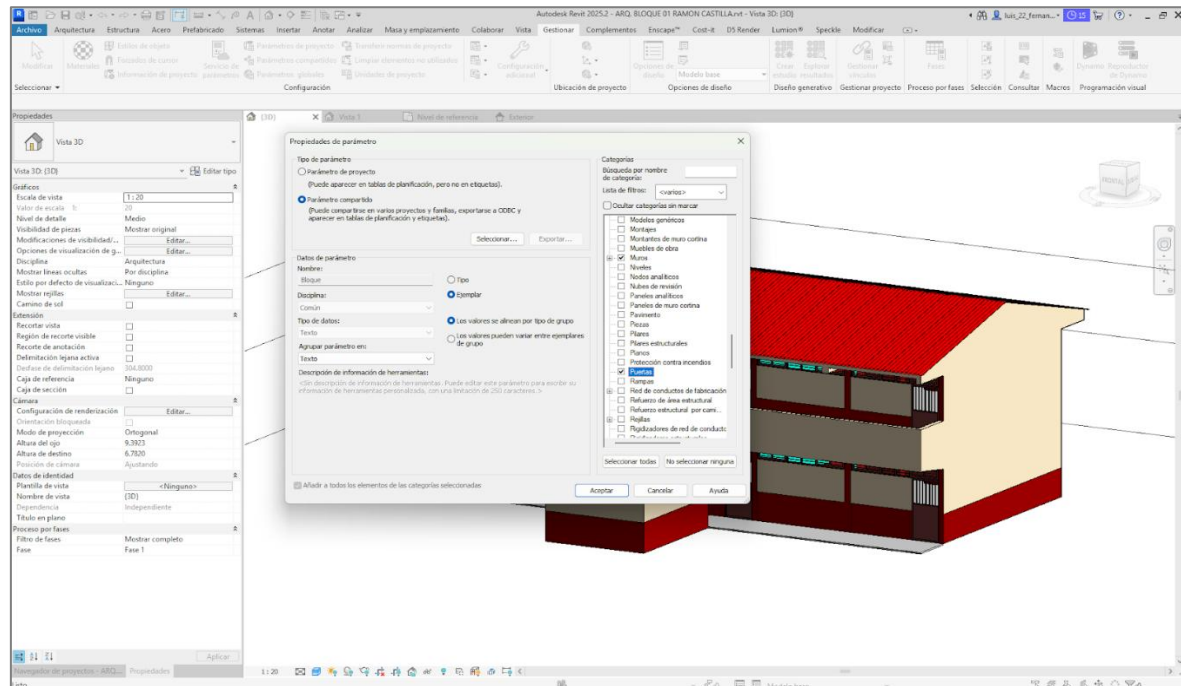


Nota. Datos tomados de Revit 2025.

La figura muestra la estructuración de los parámetros compartidos dentro del archivo definido en Revit, donde se establecieron dos grupos principales: “Datos de Identidad” y “Gestión del Mantenimiento”. El primer grupo permitió identificar y localizar los activos dentro del modelo BIM, mientras que el segundo incorporó variables técnicas, operativas, económicas y normativas necesarias para la evaluación y priorización del mantenimiento. Esta clasificación mejoró la organización de la información, facilita la interoperabilidad y optimizó la posterior extracción automatizada de datos mediante Dynamo.

Figura 101

Integración de parámetros compartidos como parámetros de proyecto y su vinculación a categorías de arquitectura, eléctricas y mobiliario



Nota. Datos tomados de Revit 2025.

La figura muestra el proceso de incorporación de los parámetros compartidos como parámetros de proyecto dentro del modelo BIM. En esta etapa, cada parámetro fue configurado definiendo su tipo de dato y asignándolo a categorías específicas correspondientes a las disciplinas de arquitectura, instalaciones eléctricas y mobiliario. Este procedimiento permitió que los elementos modelados almacenen información técnica relacionada con identificación, evaluación física, operatividad y mantenimiento, habilitando la extracción automatizada posterior mediante Dynamo y su integración con el aplicativo desarrollado.

f. Exportación de data de Revit a la aplicación para tener la base de datos de los activos en la aplicación.

Para la exportación de la data de todos los elementos modelados que componen el proyecto, se utilizó Dynamo, una extensión integrada en Autodesk Revit que permite realizar programación visual mediante nodos predefinidos. Estos nodos ejecutaron funciones específicas que, al interconectarse entre sí, permitieron automatizar procesos dentro del entorno BIM. En el presente estudio, el script desarrollado tuvo como finalidad extraer la información de cada uno de los activos modelados, capturando los parámetros definidos para

la gestión del mantenimiento y exportándolos a una base de datos estructurada para su posterior integración con el aplicativo.

g. Descripción de lo realizado en Dynamo

En el entorno de Dynamo, se desarrolló un script para extraer los parámetros de los elementos existentes en el modelo de Autodesk Revit y exportarlos a una estructura organizada tipo tabla, con el fin de generar una base de datos externa para análisis posterior (Excel / Power BI).

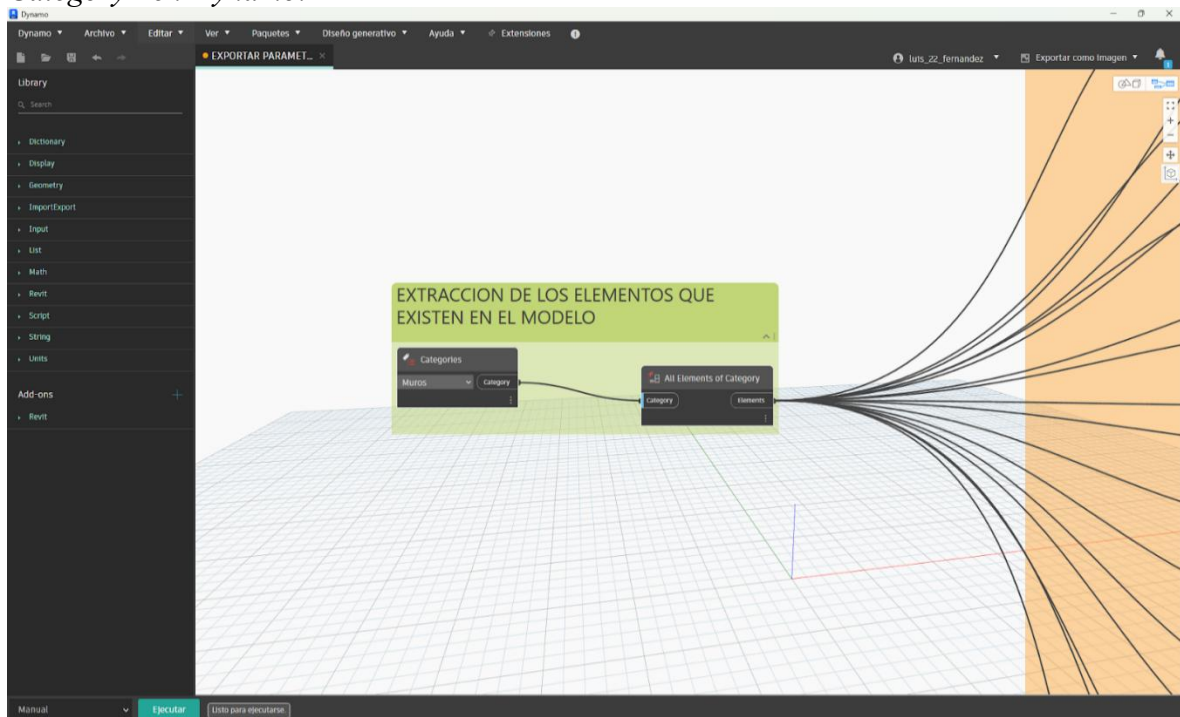
El flujo se divide en tres etapas principales:

- Extracción de elementos del modelo
- Obtención de parámetros específicos
- Creación de lista estructurada para exportación

1. Extracción de elementos del modelo

Figura 102

Extracción de elementos del modelo BIM mediante nodos “Categories” y “All Elements of Category” en Dynamo.



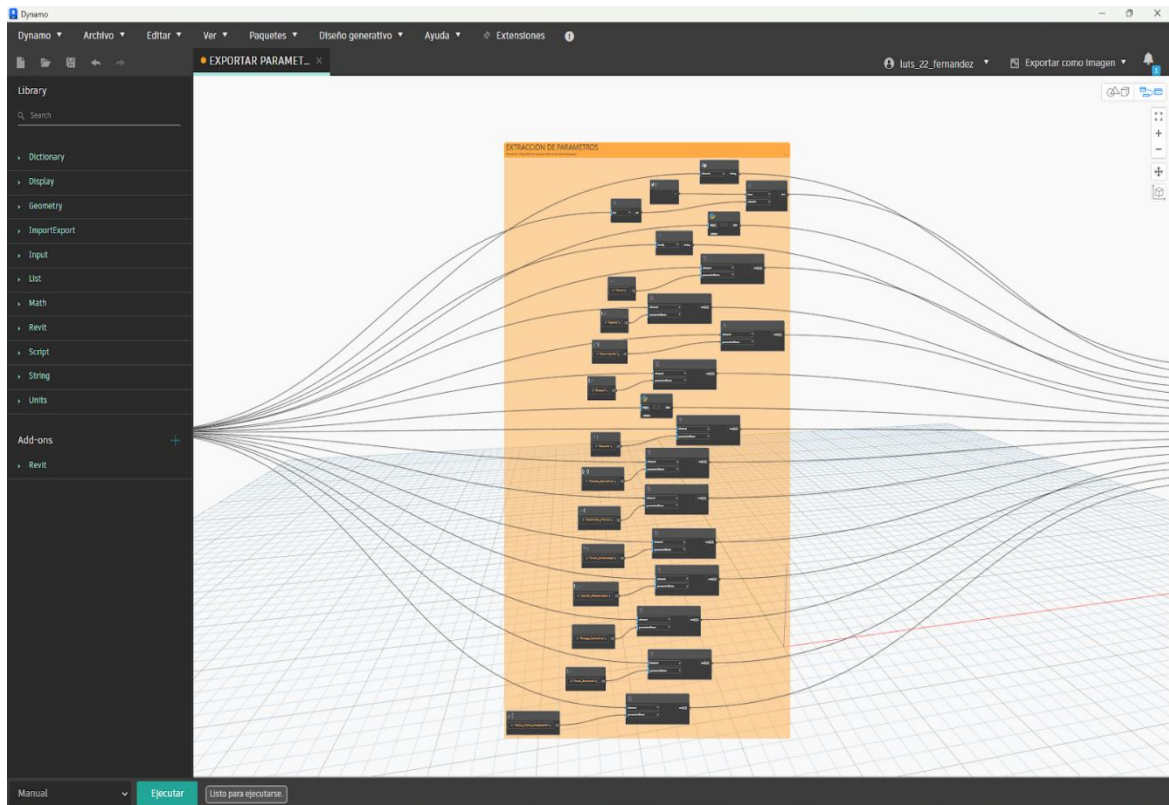
Nota. Datos tomados de Dynamo (2025).

La figura muestra la primera etapa del script desarrollado en Dynamo, correspondiente a la identificación y extracción de los elementos modelados en el entorno BIM. Para ello, se emplearon los nodos “Categories” y “All Elements of Category”, los cuales permitieron seleccionar una categoría específica y obtener todos los elementos asociados a ella. En casos particulares, se utilizó el nodo “All Elements in View” para restringir la extracción a elementos visibles en una vista determinada. Este procedimiento automatizó la recopilación masiva de activos, constituyendo el punto de partida para la posterior obtención de parámetros técnicos y su exportación a una base de datos externa.

2. Obtención de parámetros específicos

Figura 103

Configuración del script en Dynamo para la lectura y estructuración de parámetros del modelo BIM.



Nota. Datos tomados de Dynamo (2025).

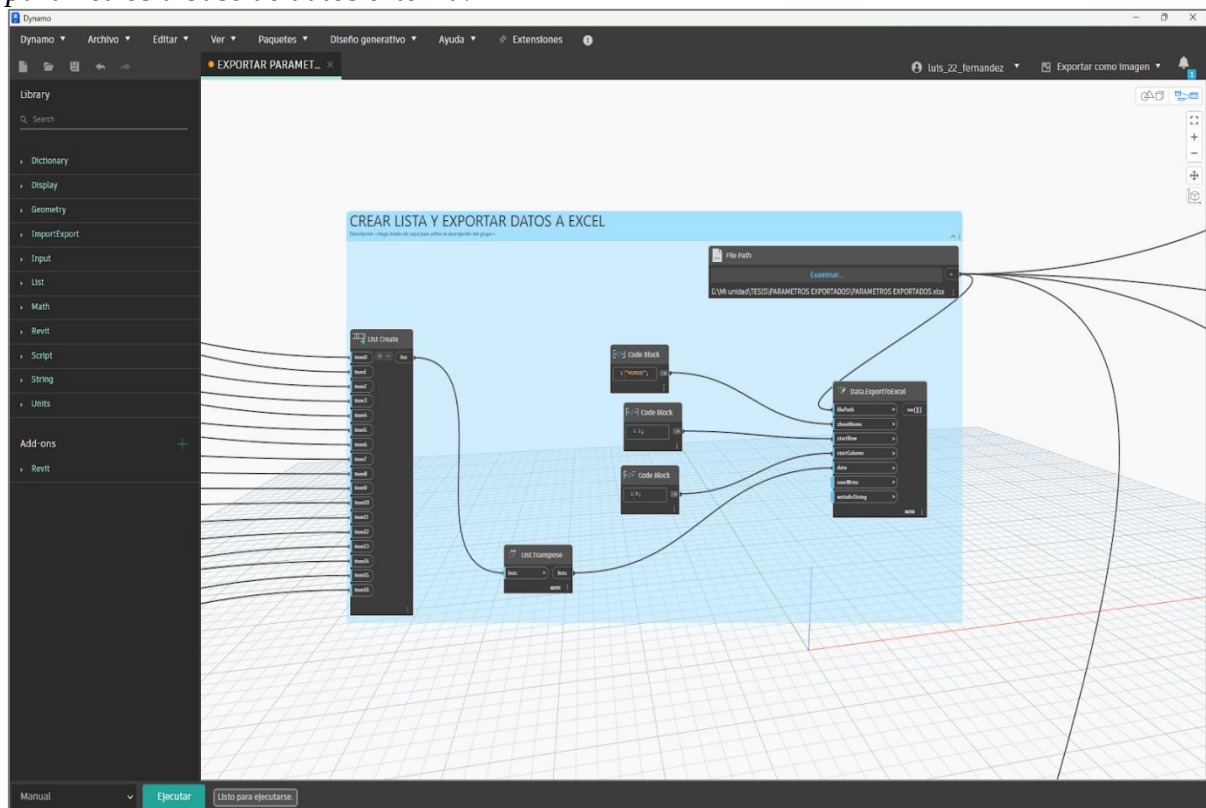
La figura muestra la segunda etapa del script desarrollado en Dynamo, correspondiente a la lectura de los parámetros asociados a cada activo del modelo BIM. Para ello, se empleó el nodo “Element.GetParameterByName”, el cual permitió capturar valores específicos definidos previamente como parámetros compartidos. Posteriormente,

mediante el uso de nodos de organización como “List.Create”, “List.Map” y “List.Transpose”, la información fue estructurada en formato tabular, facilitando su exportación a una base de datos externa. Este procedimiento transformó el modelo BIM en una fuente de datos cuantificables para la gestión del mantenimiento.

3. Creación de lista estructurada para exportación

Figura 104

Configuración del script en Dynamo para la consolidación tabular y exportación de parámetros a base de datos externa.



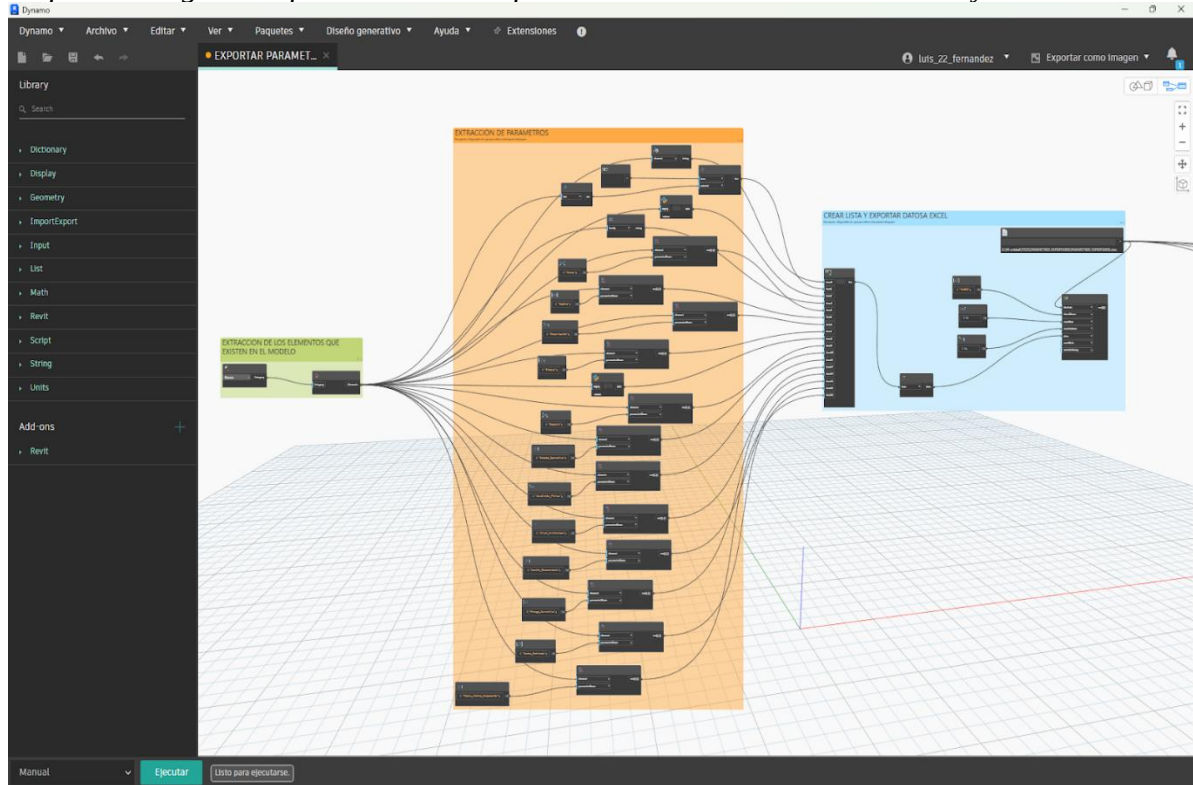
Nota. Datos tomados de Dynamo (2025).

La figura muestra la etapa final del script en Dynamo, correspondiente a la organización y estructuración de los datos extraídos del modelo BIM. Mediante el uso de nodos como “List.Flatten”, “List.Clean” y “List.Combine”, la información fue depurada y consolidada en una estructura tabular coherente. Posteriormente, a través del nodo “Data.ExportExcel”, se generó un archivo en formato Excel que contiene los parámetros técnicos de todos los activos modelados. Este proceso permitió convertir el modelo BIM en

una base de datos externa estructurada, garantizando interoperabilidad entre Revit, Excel y el aplicativo de gestión del mantenimiento.

Figura 105

Esquema integral del proceso de interoperabilidad BIM-Excel mediante Dynamo

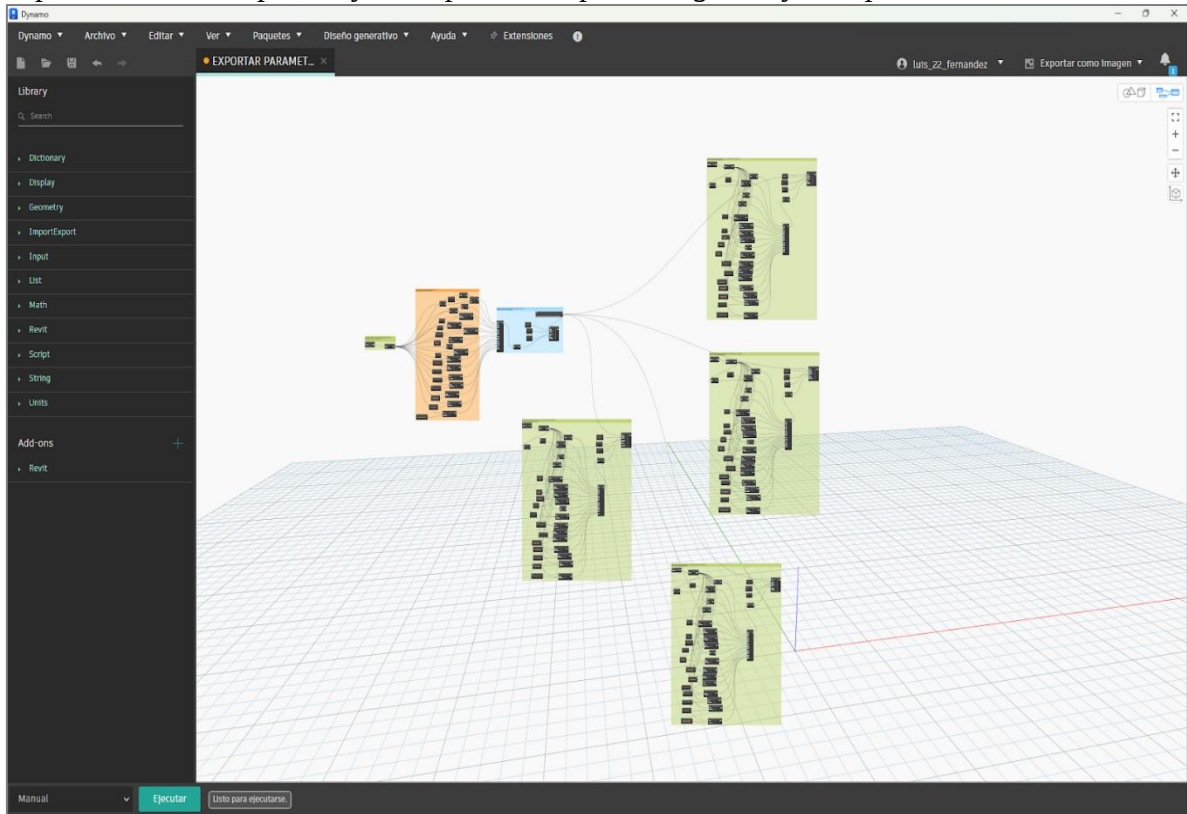


Nota. Datos tomados de Dynamo (2025).

La figura presenta el esquema integral del script desarrollado en Dynamo, el cual permitió automatizar la extracción masiva de parámetros asociados a los activos modelados en el entorno BIM. El proceso integró la selección de elementos, la lectura de parámetros específicos y la estructuración tabular de los datos para su exportación a formato Excel. Entre los parámetros extraídos se encuentran *Accion_Recomendada*, *Condicion_Fisica*, *Costo_Estimado*, *Estado_Operativo*, *Fecha_Ultima_Inspeccion*, *Nivel_Criticidad* y *Riesgo_Normativo*, los cuales constituyen la base técnica para la evaluación y priorización del mantenimiento. Este procedimiento transformó el modelo BIM en una base de datos estructurada, permitiendo su integración con el aplicativo desarrollado y consolidando la interoperabilidad digital del sistema.

Figura 106

Replicación del script en Dynamo para múltiples categorías y disciplinas del modelo BIM.



Nota. Datos tomados de Dynamo (2025).

La figura muestra la adaptación del script desarrollado en Dynamo para trabajar con múltiples categorías del modelo BIM. Debido a que la entrada del flujo permite procesar una sola categoría por ejecución, fue necesario replicar el procedimiento para cada grupo de elementos que requería exportación. Este proceso se ejecutó de manera independiente en los tres modelos disciplinarios (arquitectura, instalaciones eléctricas y mobiliario), garantizando la extracción integral de los parámetros definidos para la gestión del mantenimiento. La replicabilidad del script demostró la escalabilidad del sistema y su aplicabilidad a distintos tipos de activos dentro del entorno BIM.

Figura 107

Archivos Excel generados tras la ejecución del script de extracción de parámetros en Dynamo.

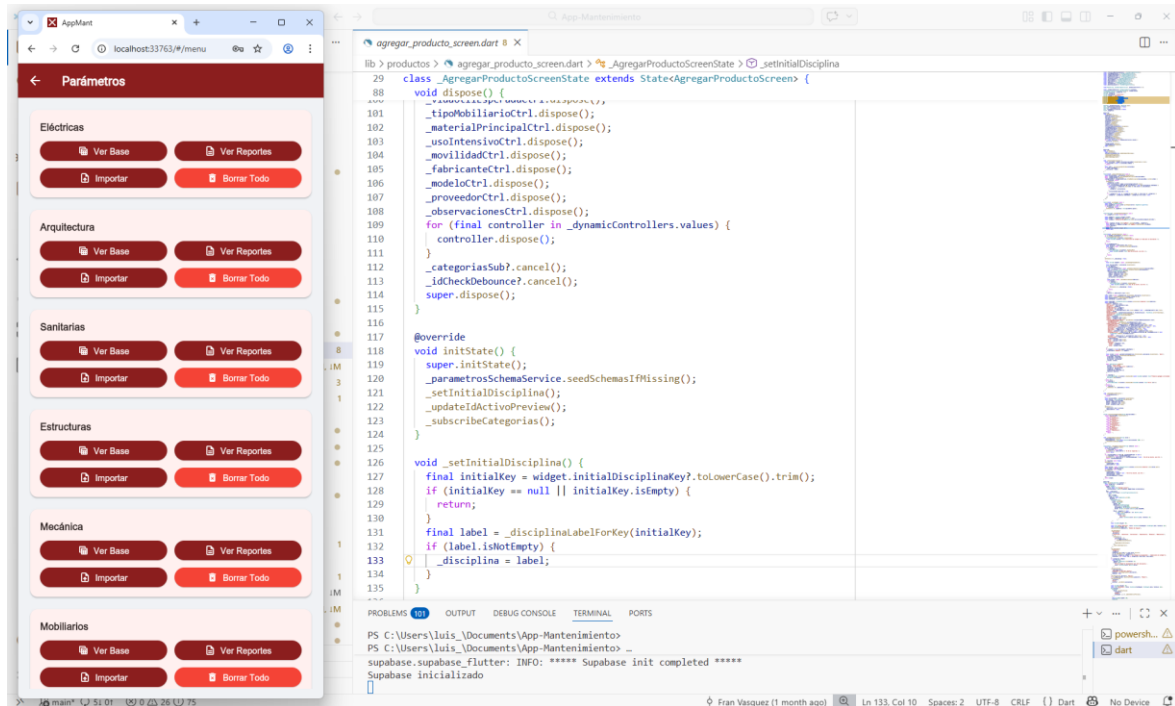
ID	Activo	Disciplina	Categoría	A	Tipo	Activo	Marca	Modelo	Descripción
1	34cfa9b8-281	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
2	06b3d948-62	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
3	11e48f8f-62	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
4	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
5	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
6	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
7	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
8	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
9	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
10	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
11	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
12	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
13	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
14	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
15	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
16	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
17	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
18	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
19	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
20	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
21	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
22	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
23	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
24	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
25	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
26	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
27	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
28	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
29	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
30	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
31	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
32	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
33	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
34	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
35	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
36	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
37	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
38	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
39	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
40	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
41	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
42	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	
43	4306b6d9-e6	MOBILIARIO	Dispositivos	TV 51"	S/N	S/N	TV 51"	BL	

Nota. Datos tomados de Excel (2025).

La figura muestra los archivos Excel generados automáticamente tras la ejecución del script desarrollado en Dynamo. Cada archivo contiene la información estructurada de los activos modelados en las distintas disciplinas del proyecto, incluyendo parámetros técnicos, operativos y económicos necesarios para la gestión del mantenimiento. La generación de estos archivos permitió transformar el modelo BIM en una base de datos externa organizada, la cual fue posteriormente importada al aplicativo desarrollado. Asimismo, la posibilidad de reejecutar el script garantizó la actualización constante de la información, asegurando coherencia entre el modelo BIM y el sistema de gestión.

Figura 108

Importación de la base de datos Excel al aplicativo de gestión de mantenimiento.



Nota. Datos tomados de Aplicativo (2026).

La figura muestra el módulo de importación del aplicativo desarrollado, donde se cargaron los archivos Excel previamente generados desde el modelo BIM mediante Dynamo. Cada disciplina cuenta con un apartado independiente que permite importar, visualizar y gestionar la base de datos correspondiente. Este procedimiento permitió integrar la información técnica de los activos al sistema digital de mantenimiento, consolidando la interoperabilidad entre el entorno BIM y el aplicativo. La sincronización de datos posibilitó la actualización constante del estado de los activos y facilita la toma de decisiones basada en información estructurada.

Figura 109

Proceso de análisis e importación de base de datos Excel en el aplicativo de mantenimiento.

← Importar Base (Excel)

Disciplina: Arquitectura

Seleccionar archivo (.xlsx)

Archivo: PARAMETROS EXPORTADOS ARQUITECTURA.xlsx

☐ Sobrescribir campos vacios

Simular importacion (no guardar) ☒

Analizar archivo

Resumen

Total filas: 754
Creados: 0
Actualizados: 754
Omitidos: 0
Categorías a crear: 0

Cambios por fila ▼

Confirmar e Importar

Nota. Datos tomados de Aplicativo (2026).

La figura muestra el proceso de carga y validación del archivo Excel exportado desde el modelo BIM. El sistema permitió analizar previamente el archivo, visualizar un resumen de registros procesados y confirmar la importación definitiva. En el caso presentado, se procesaron 754 registros correspondientes a la disciplina de Arquitectura, actualizándose la totalidad de los activos existentes sin generar duplicidades. Este mecanismo asegura

consistencia de datos y sincronización entre el entorno BIM y el aplicativo de gestión del mantenimiento.

2.3.4.5. FASE 5: Implementación y Evaluación del Aplicativo

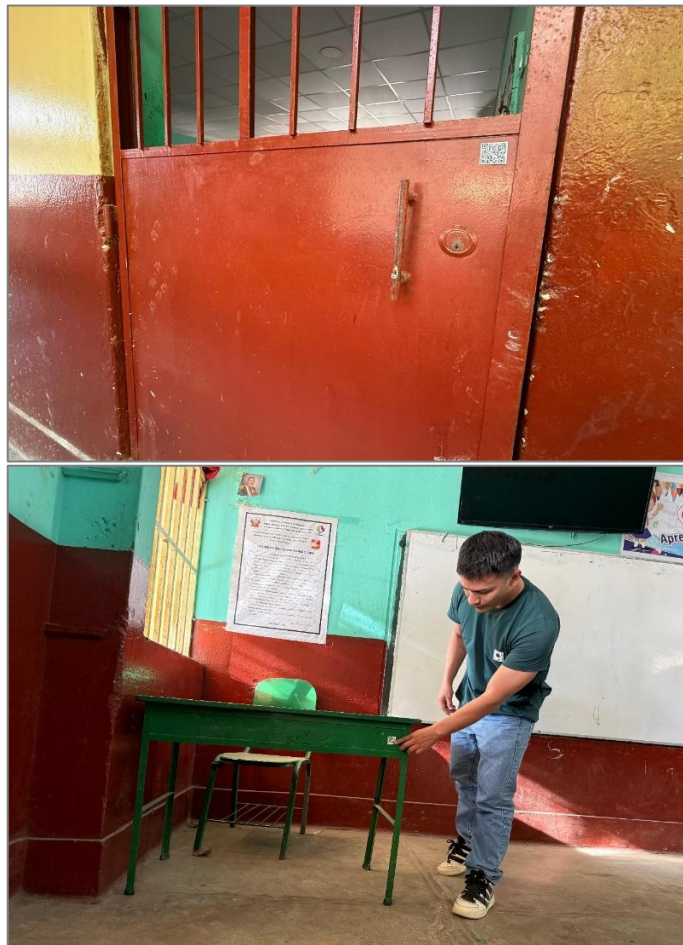
Figura 110

Inicio del levantamiento en campo con el aplicativo



Nota. Elaboración propia

La figura muestra el inicio del proceso de levantamiento de información en campo utilizando el aplicativo desarrollado para la gestión y mantenimiento de infraestructura educativa. En la imagen se observa a un usuario dentro de un aula de la Institución Educativa Ramón Castilla y Marquesado, mostrando en un dispositivo móvil la interfaz principal del aplicativo, la cual permitió acceder a diferentes módulos funcionales del sistema.

Figura 111*Identificación de los activos mediante código QR*

Nota. Elaboración propia

Las figuras muestran el proceso de identificación y registro de activos de infraestructura dentro de la Institución Educativa Ramón Castilla y Marquesado mediante la implementación de códigos QR. Se identificaron diferentes elementos del entorno educativo, como puertas, mobiliario escolar y equipos tecnológicos, a los cuales se les asignó un código QR único. La colocación de estos códigos permitió vincular cada activo físico con un registro digital dentro del aplicativo desarrollado, lo que facilitó su identificación, seguimiento y gestión dentro del sistema de mantenimiento. A través de este mecanismo, cada elemento de la infraestructura pudo ser escaneado mediante un dispositivo móvil, permitiendo acceder rápidamente a su información técnica, estado de conservación, historial de mantenimiento y posibles incidencias. En este sentido, la identificación mediante códigos QR constituyó una herramienta tecnológica que fortaleció

el control y monitoreo de los elementos de infraestructura, permitiendo una gestión más eficiente, trazable y sistematizada de los activos dentro de la institución educativa.

Figura 112

Generación de reporte de cada activo mediante el aplicativo



Nota. Elaboración propia

La figura muestra el proceso de generación y consulta de información de un activo de mobiliario mediante el uso del aplicativo móvil desarrollado para la gestión de infraestructura. En la imagen se observa un dispositivo móvil escaneando el código QR colocado en la parte posterior de una silla escolar, lo cual permite identificar de manera inmediata el activo dentro del sistema. A través del escaneo del código QR, el aplicativo accedió al registro digital del mobiliario, permitiendo visualizar y registrar información relacionada con sus características, estado de conservación y necesidades de mantenimiento. Este procedimiento facilitó la generación de reportes específicos para cada activo, permitiendo documentar incidencias, registrar observaciones y mantener un historial actualizado de su estado. Asimismo, el uso de esta herramienta tecnológica optimizó el proceso de levantamiento de información en campo, ya que permitió vincular directamente el activo físico con su base de datos digital dentro del sistema de gestión. De esta manera, se mejoró la trazabilidad de los elementos de infraestructura y se fortaleció la toma de decisiones relacionadas con la planificación y ejecución de actividades de mantenimiento.

Figura 113*Generación de reportes*

1:54 85%

< **Generar Reporte**

El número de reporte será asignado automáticamente al guardar.

Tipo de Reporte*
Inspección

Activo*
tarrajeo techo e=1.5 cm 2

Ubicación
Bloque: BLOQUE - Nivel: Nivel 2

Categoría

y de con

q w e r t y u i o p

a s d f g h j k l ñ

↑ z x c v b n m ↵

123 espacio intro

😊 🎤

Nota. Datos tomados de Aplicativo (2026).

La figura muestra la interfaz del aplicativo móvil utilizada para la generación de reportes durante el proceso de inspección de la infraestructura educativa. En la pantalla del dispositivo se observa el formulario denominado “Generar Reporte”, el cual permite registrar información relacionada con el estado de los activos evaluados. Dentro del formulario se incluyen campos como el tipo de reporte, el activo inspeccionado y su ubicación dentro de la infraestructura, lo que permitió asociar cada registro con un elemento específico del sistema constructivo. En el ejemplo presentado, el reporte corresponde a una inspección del elemento “tarrajeo del techo”, ubicado en el bloque y

nivel correspondiente dentro de la institución educativa. Este módulo del aplicativo permitió documentar de manera estructurada las observaciones realizadas durante el levantamiento en campo, facilitando el registro de incidencias, el seguimiento del estado de los elementos constructivos y la generación de información para la gestión del mantenimiento.

Figura 114

Guardado de reportes generados

Nota. Datos tomados de Aplicativo (2026).

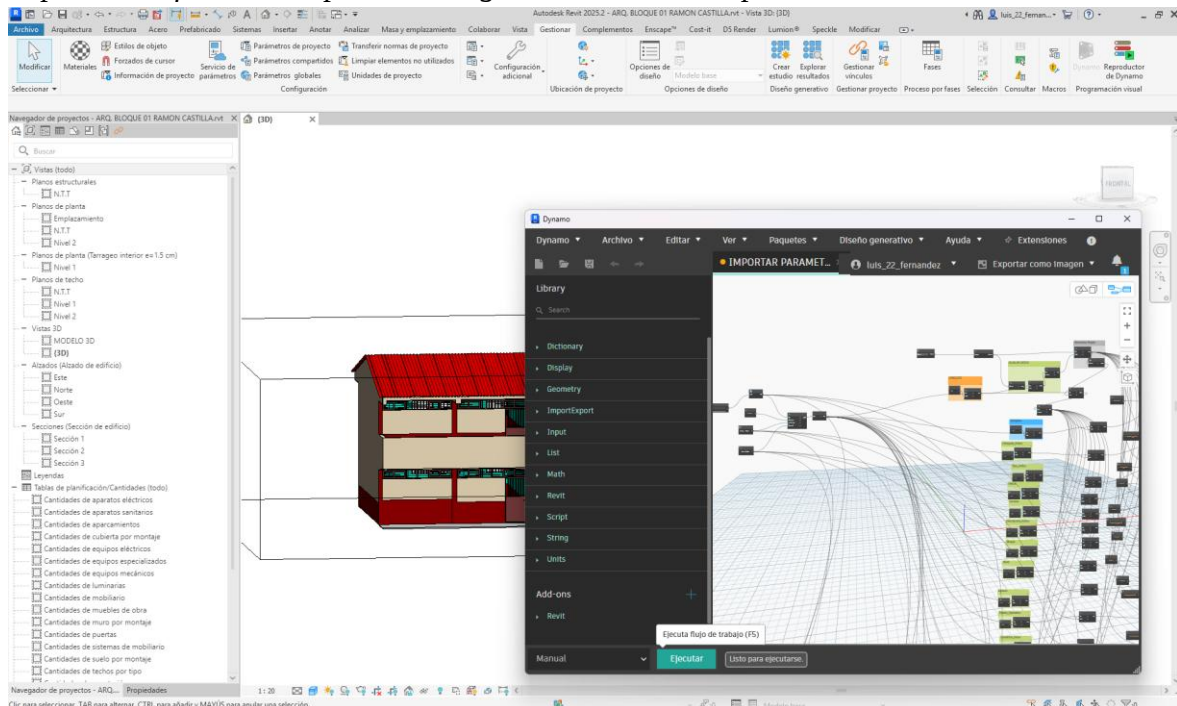
La figura muestra la interfaz del aplicativo móvil durante la etapa final del registro de un reporte de inspección, donde se procede al guardado de la información recopilada en

campo. En la pantalla se observan los diferentes campos que permiten completar el registro del reporte, incluyendo el impacto de la falla, el nivel de riesgo normativo, las acciones tomadas o comentarios técnicos, así como la incorporación de fotografías como evidencia del estado del elemento evaluado. El sistema permitió documentar de manera detallada las condiciones observadas en los activos de infraestructura, registrando información relevante como el tipo de intervención requerida y las observaciones realizadas durante la inspección. Asimismo, la posibilidad de adjuntar imágenes facilita la validación visual de las condiciones detectadas, fortaleciendo la trazabilidad y verificación de la información registrada. Una vez completados los campos del formulario, el aplicativo permitió guardar el reporte generado, almacenando los datos dentro del sistema de gestión para su posterior consulta, análisis y seguimiento. Este proceso contribuye a mantener un registro digital organizado de las inspecciones realizadas, permitiendo disponer de información actualizada sobre el estado de los activos y apoyar la toma de decisiones relacionadas con el mantenimiento de la infraestructura.

2.3.4.6. FASE 6: Integración del Aplicativo con el Modelo BIM

Figura 115

Importación final de los parámetros gestionados con el aplicativo.

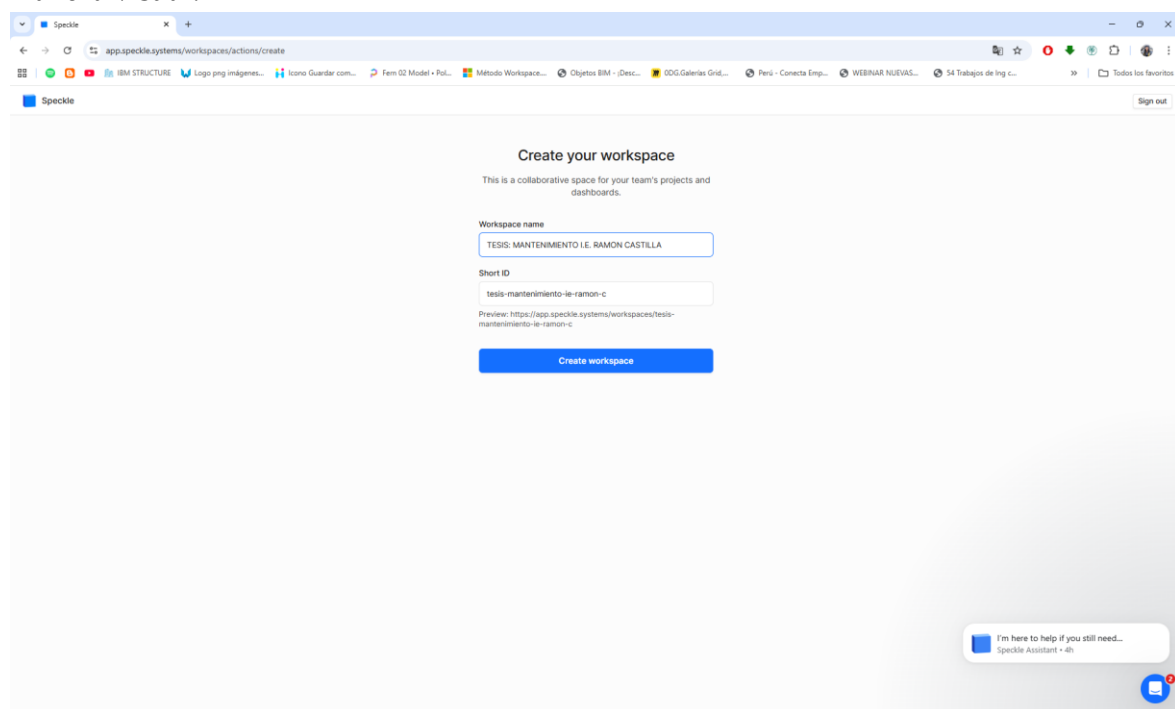


Nota. Datos tomados de BIM (2026).

La figura muestra el proceso final de integración de la información gestionada mediante el aplicativo móvil con el modelo BIM desarrollado en Revit. En la imagen se observa el entorno de trabajo de Autodesk Revit, donde se encuentra abierto el modelo digital de la infraestructura educativa, junto con la interfaz de Dynamo, utilizada para automatizar la importación de los datos provenientes de la base de datos generada en el aplicativo. El flujo de trabajo implementado en Dynamo permite leer el archivo de datos exportado previamente desde el aplicativo en formato Excel y vincular la información con los elementos correspondientes del modelo BIM. A través de este procedimiento se actualizan automáticamente los parámetros asociados a cada activo del modelo, tales como Estado Operativo, Condición Física, Nivel de Criticidad, Riesgo Normativo, Acción Recomendada, Costo Estimado y Fecha de Última Inspección. Este proceso de importación automatizada permitió establecer un vínculo entre la información recolectada en campo y el modelo digital de la infraestructura, facilitando la actualización dinámica de los parámetros de mantenimiento dentro del entorno BIM. De esta manera, el modelo se convierte en una herramienta de gestión que integra tanto la representación geométrica como la información técnica relacionada con el estado y mantenimiento de los activos.

Figura 116

Creación del proyecto en speckle, herramienta que permitirá gestionar la información de manera visual.

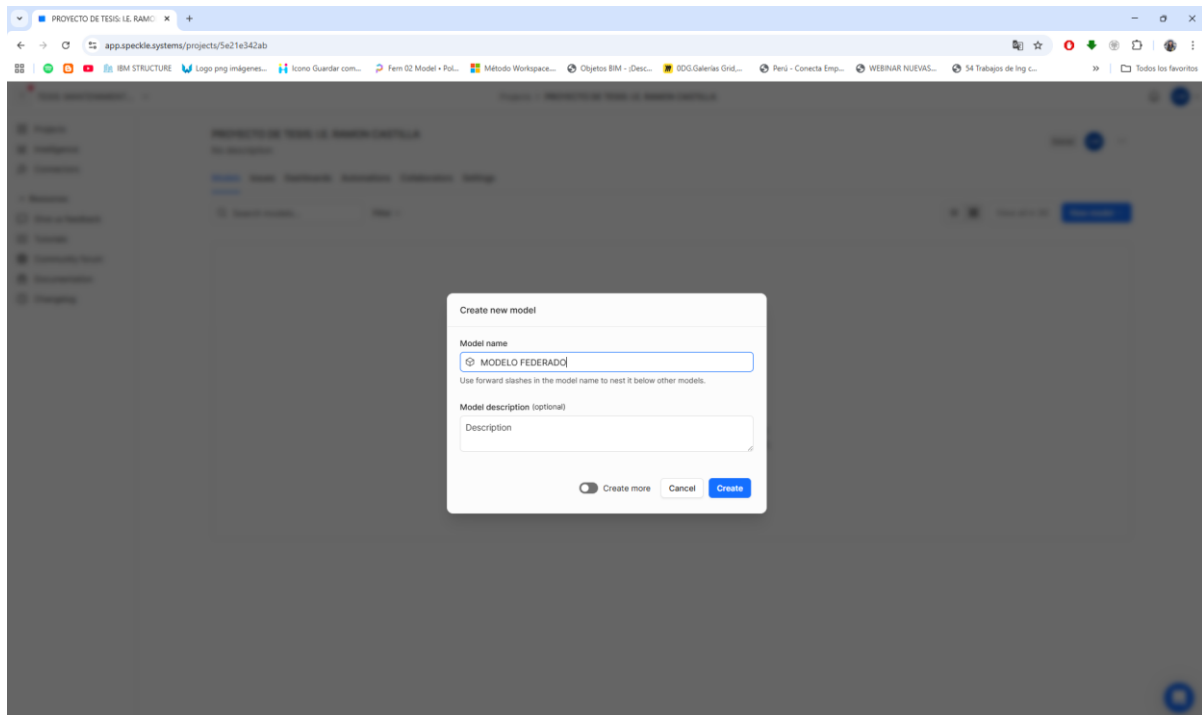


Nota. Datos tomados de Speckle (2026).

La imagen muestra la interfaz de creación de un espacio de trabajo (workspace) en la plataforma Speckle, utilizada para la gestión e integración de información dentro del entorno digital del proyecto. En este proceso se configuró el espacio denominado “TESIS: MANTENIMIENTO I.E. RAMÓN CASTILLA”, el cual permitió centralizar y organizar los datos generados durante el desarrollo del aplicativo. La creación de este espacio de trabajo permitió establecer un entorno colaborativo donde se almacenaron y gestionaron los datos relacionados con los activos de infraestructura identificados en la institución educativa. Asimismo, Speckle funcionó como un repositorio de información que facilitó la conexión y visualización de los datos dentro de herramientas de análisis como Power BI, permitiendo representar de manera gráfica y dinámica la información recopilada. En este sentido, la implementación de Speckle permitió estructurar y administrar los datos del sistema de mantenimiento de forma interoperable, contribuyendo a mejorar la visualización, análisis y toma de decisiones en la gestión de la infraestructura educativa.

Figura 117

Plataforma Speckle durante el proceso de creación de un nuevo modelo dentro del proyecto BIM.



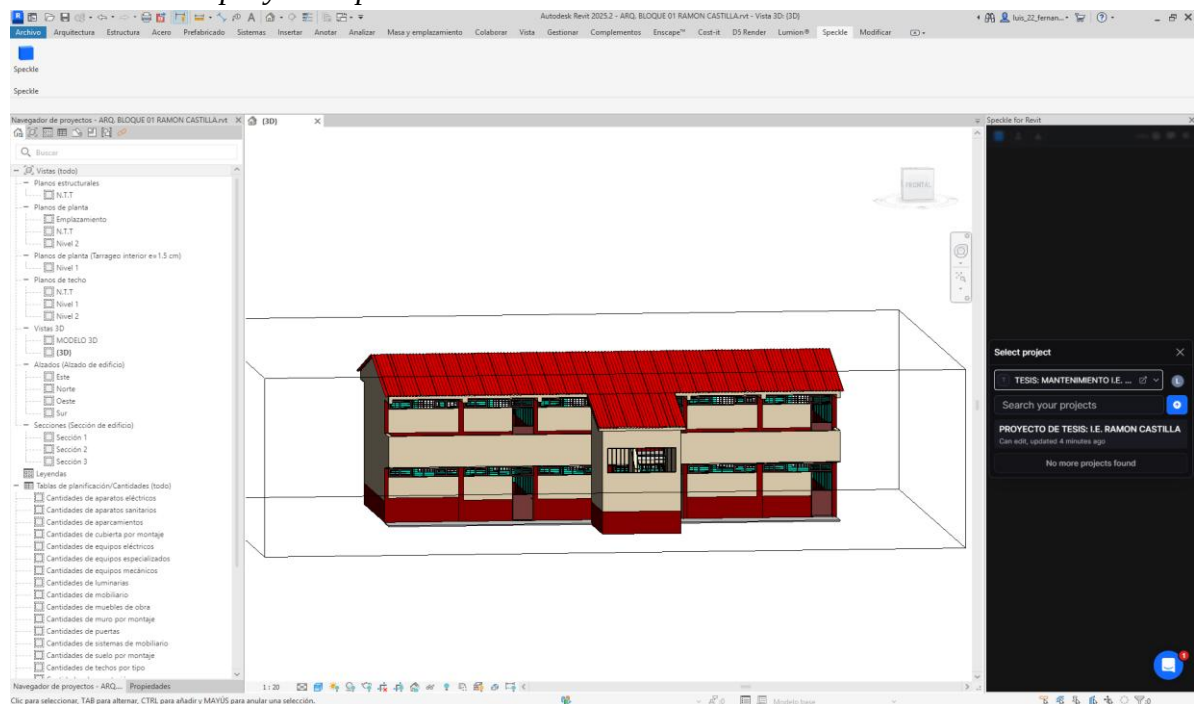
Nota. Datos tomados de Speckle (2026).

La figura muestra la interfaz de la plataforma Speckle durante el proceso de creación de un nuevo modelo dentro del proyecto BIM desarrollado para la infraestructura educativa.

En la ventana emergente se observa la opción “Create new model”, donde se define el nombre del modelo denominado “Modelo Federado”, el cual sirvió como contenedor principal de la información generada durante el desarrollo del proyecto. El modelo federado tiene como finalidad integrar en un único entorno la información proveniente de las diferentes disciplinas y fuentes de datos del proyecto. En este caso, el modelo consolidado tanto la información geométrica correspondiente al modelo tridimensional de la infraestructura (modelo 3D) como los datos asociados a los parámetros de los elementos, los cuales han sido previamente actualizados a partir de la información recolectada mediante el aplicativo de inspección y mantenimiento. La creación de este modelo federado permitió centralizar la información del proyecto dentro de una plataforma colaborativa, facilitando la visualización, gestión e intercambio de datos entre diferentes herramientas y usuarios. Asimismo, esta integración permitió mantener vinculados los datos de mantenimiento registrados en campo con los elementos del modelo BIM, fortaleciendo la interoperabilidad entre el sistema de gestión de activos, el modelo BIM desarrollado en Revit y la plataforma de intercambio de información.

Figura 118

Vinculación del proyecto Speckle con el modelo BIM en Revit

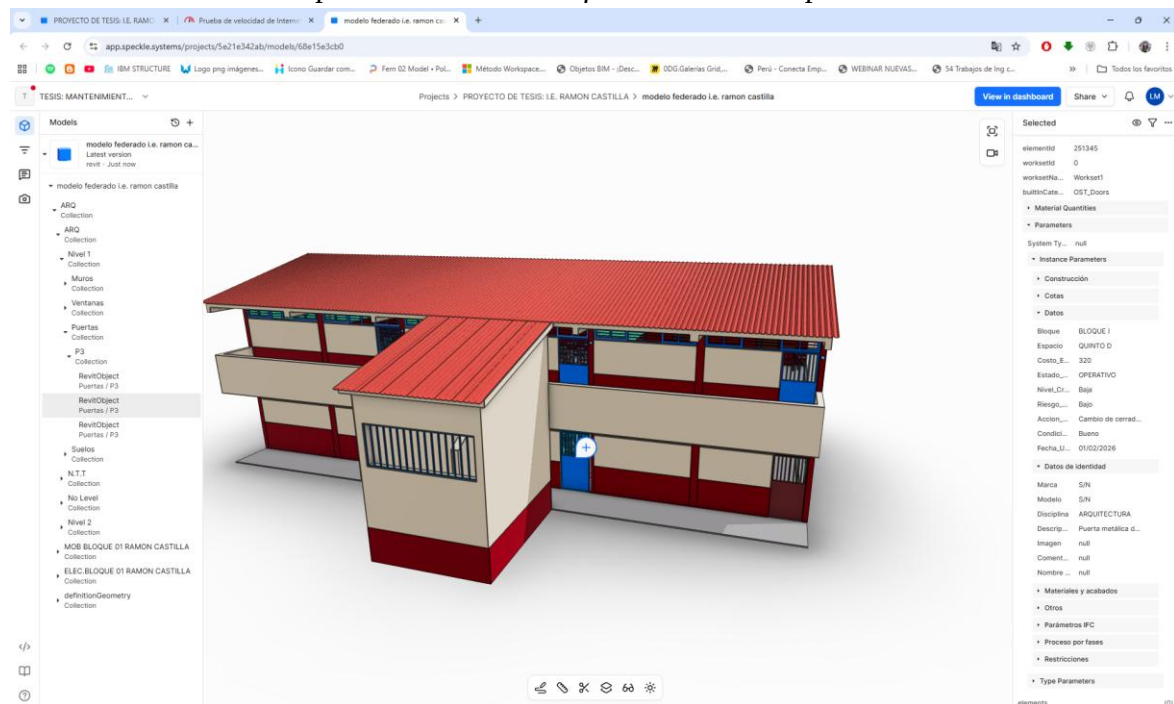


Nota. Datos tomados de BIM(2026).

La figura muestra el entorno de trabajo de Autodesk Revit junto con el complemento de Speckle, el cual permite la conexión entre el modelo BIM y la plataforma de gestión de datos utilizada en el proyecto. En la interfaz se observa el modelo tridimensional de la infraestructura educativa y, en el panel lateral derecho, la ventana de Speckle donde se visualiza el proyecto previamente creado en la plataforma. Una vez creado el proyecto en Speckle, este se sincroniza automáticamente con el entorno de Revit mediante el complemento correspondiente, permitiendo que el proyecto sea identificado y seleccionado directamente desde el modelo BIM. Esta integración facilita la conexión entre ambas plataformas, habilitando el envío y la recepción de información asociada al modelo. A través de esta vinculación, el modelo desarrollado en Revit puede exportar tanto la información geométrica del modelo tridimensional como los parámetros asociados a los elementos, los cuales han sido previamente actualizados a partir de los datos recolectados mediante el aplicativo de inspección y mantenimiento. De esta manera, los elementos del modelo BIM pudieron compartir su información con la plataforma Speckle, permitiendo su visualización, análisis y gestión en un entorno interoperable.

Figura 119

Modelo vinculado con speckle con toda la información de los parámetros

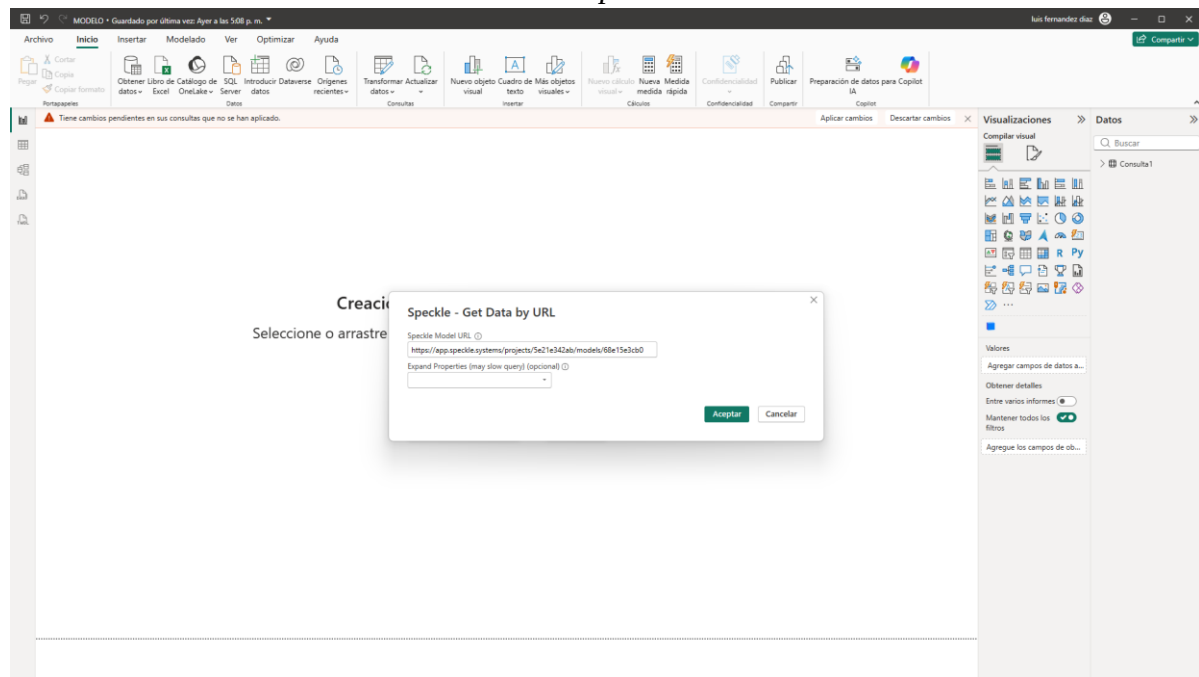


Nota. Datos tomados de BIM (2026).

La figura muestra la visualización del modelo BIM federado dentro de la plataforma Speckle, donde se integra tanto la representación geométrica tridimensional de la infraestructura educativa como la información asociada a los parámetros de los elementos del modelo. En la interfaz se observa el modelo 3D del bloque de la institución educativa, el cual ha sido previamente desarrollado en Revit y posteriormente vinculado a la plataforma mediante el proceso de interoperabilidad implementado. En el panel lateral de la plataforma se pueden visualizar los diferentes componentes del modelo organizados jerárquicamente, tales como niveles, muros, puertas y otros elementos constructivos. Asimismo, al seleccionar un elemento específico del modelo, la plataforma permitió visualizar los parámetros asociados, los cuales incluyen información técnica como bloque, nivel, estado operativo, condición física, nivel de criticidad, riesgo normativo, acción recomendada, costo estimado y fecha de última inspección. Estos parámetros han sido previamente gestionados y actualizados mediante el aplicativo desarrollado para la recolección de datos en campo, lo que permite integrar la información de mantenimiento con el modelo digital de la infraestructura. De esta manera, el modelo BIM no solo representa la geometría del edificio, sino que también funciona como un contenedor de información técnica relacionada con la gestión y mantenimiento de los activos.

Figura 120

Vinculación de datos del modelo BIM desde Speckle hacia Power BI



Nota. Datos tomados de BIM (2026).

La figura muestra el proceso de conexión entre la plataforma Speckle y Power BI, mediante el cual se realiza la importación de la información del modelo BIM para su posterior análisis y visualización. En la interfaz se observa la ventana denominada “Speckle – Get Data by URL”, donde se introdujo la dirección del modelo alojado en Speckle para acceder a los datos asociados al proyecto. A través de esta conexión, Power BI puede recuperar directamente la información almacenada en el modelo federado, incluyendo los parámetros asociados a los elementos de la infraestructura que han sido previamente actualizados mediante el aplicativo de inspección y mantenimiento. Estos datos comprenden variables como estado operativo, condición física, nivel de criticidad, riesgo normativo, acciones recomendadas, costos estimados y fechas de inspección, entre otros. La vinculación de Power BI con Speckle permitió transformar la información técnica contenida en el modelo BIM en conjuntos de datos estructurados que pueden ser utilizados para el análisis y la generación de reportes interactivos. De esta manera, se facilita la creación de visualizaciones y paneles de control que permitieron evaluar el estado de los activos, identificar elementos críticos y apoyar la toma de decisiones relacionadas con la gestión y mantenimiento de la infraestructura.

Figura 121

Extracción de los parámetros a gestionar en power BI

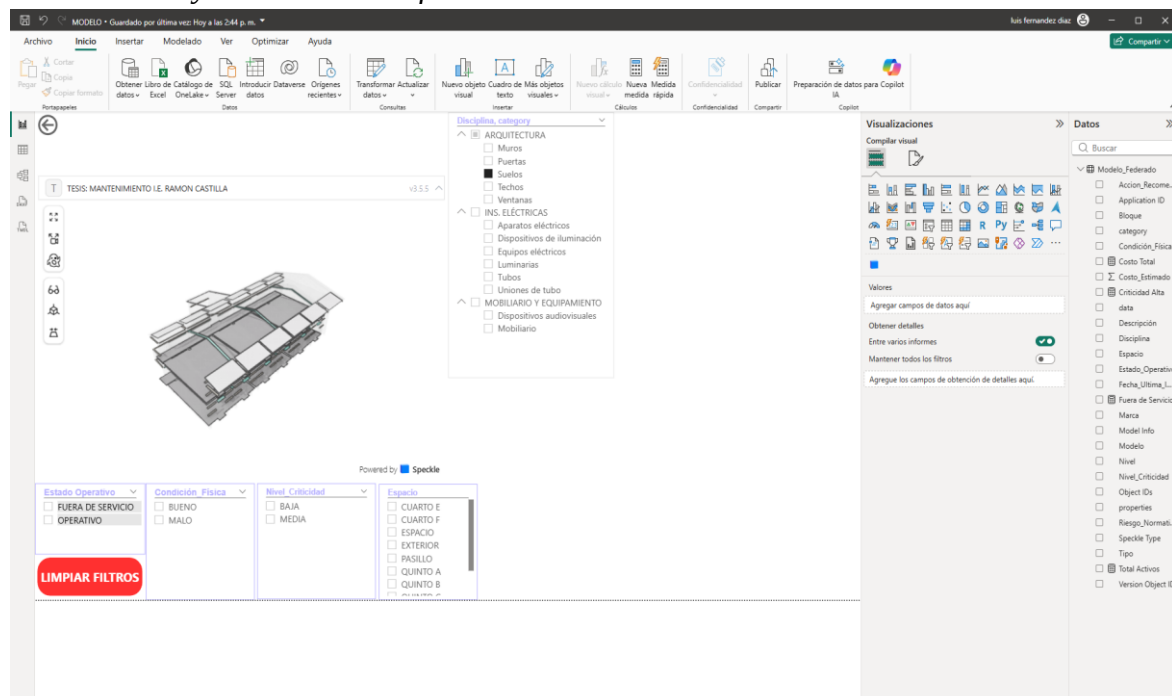
ID	Application	data	properties
1	AC002/mn/Nt12	sec88a0d-0d07-4e09-45af-ba72212a62-000888a_180ee	Record
2	AC002/mn/Nt12	715a0074-66e7-4d06-9805-4225daa3e7-0003951	Record
3	AC002/mn/Nt12	91a3237e-9a23-4111-838b-64a00d5d907-000959a_180ee	Record
4	AC002/mn/Nt12	9d9d6af-6a5d-4e0e-4d3a-701320a4c8-000959a_180ee	Record
5	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
6	AC002/mn/Nt12	af42123b-e176-4e84-9542-5a2029a20a-0003158a	Record
7	AC002/mn/Nt12	683232b-108c-4a22-9a80-1749e198c11-00030de	Record
8	AC002/mn/Nt12	1a418b6-5041-4c5a-404f-c599869a9-000358e	Record
9	AC002/mn/Nt12	3a631a70-4010-4089-4945-0113b-0004049	Record
10	AC002/mn/Nt12	af42123b-e176-4e84-9542-5a2029a20a-0003158a	Record
11	AC002/mn/Nt12	c0b3a89e-df16-4e08-9a05-913eeef26-000495c_180ee	Record
12	AC002/mn/Nt12	8056a366-7335-4893-4357-4926154d45-00030875	Record
13	AC002/mn/Nt12	4ee47397-4849-4090-8887-1444a0d89a-000347af	Record
14	AC002/mn/Nt12	00f9a30f-7722-4317-4246-8c10204a87-0004010_180ee	Record
15	AC002/mn/Nt12	09191875-1680-4e0d-95af-131a0a0b0c-00044890	Record
16	AC002/mn/Nt12	00442367-ac33-4713-8333-07735a706a00-0008a07_180ee	Record
17	AC002/mn/Nt12	20a3a70-0795-4805-0aee-077114057a1-00090a5_180ee	Record
18	AC002/mn/Nt12	9eaf985-9d7e-4009-47ef-0a5c5a24a5a-000479e7_180ee	Record
19	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
20	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
21	AC002/mn/Nt12	13f6a45-23a3-460c-8a5a-13a05513a055-000289a2	Record
22	AC002/mn/Nt12	c18ca11-1100-4910-ea03-27c86a4380-00099a5_180ee	Record
23	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
24	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
25	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
26	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
27	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
28	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
29	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
30	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
31	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
32	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
33	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
34	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
35	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
36	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
37	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
38	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
39	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
40	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
41	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
42	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
43	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
44	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
45	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
46	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
47	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
48	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
49	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record
50	AC002/mn/Nt12	af7500ea-dc11-4077-97b3-6384a16a06-0003951	Record

Nota. Datos tomados de power bi (2026).

La figura muestra el entorno del Editor de Power Query en Power BI, donde se realizó el proceso de extracción y transformación de los datos provenientes del modelo BIM vinculado a través de la plataforma Speckle. En la interfaz se observa una tabla que contiene múltiples registros correspondientes a los elementos del modelo, junto con sus propiedades y parámetros asociados. En esta etapa del proceso, los datos obtenidos desde Speckle son organizados y estructurados mediante las herramientas de transformación de Power Query. A través de este procedimiento se expandieron y seleccionan los campos de interés, permitiendo identificar y aislar los parámetros que fueron utilizados para el análisis y la gestión de la información dentro de Power BI. Entre los datos extraídos se encuentran atributos relacionados con la clasificación de los elementos, categorías del modelo, propiedades técnicas y otros parámetros asociados a la infraestructura. Este proceso de transformación permitió convertir la información proveniente del modelo BIM en un conjunto de datos estructurados, facilitando su posterior análisis mediante tablas, indicadores y visualizaciones.

Figura 122

Visualización y análisis de la información del modelo BIM en Power BI



Nota. Datos tomados de BIM (2026).

La figura muestra el panel de visualización desarrollado en Power BI, donde se presenta la información del modelo BIM integrada con los datos de mantenimiento obtenidos

a partir del aplicativo de inspección y gestión de activos. En esta interfaz se utilizaron diferentes herramientas de visualización como segmentadores de datos, tablas, tarjetas de información y gráficos, los cuales permitieron analizar y filtrar la información registrada en el sistema. Asimismo, se incorpora un visualizador tridimensional del modelo BIM proporcionado por Speckle, el cual permitió interactuar directamente con el modelo dentro del entorno de Power BI. Este visor facilita la identificación espacial de los elementos de la infraestructura y su relación con los parámetros asociados, permitiendo comprender de manera visual el estado y condición de los activos. Los segmentadores de datos permiten filtrar la información según criterios como estado operativo, condición física, nivel de criticidad y ubicación dentro de la infraestructura, facilitando el análisis específico de los elementos que presentan determinadas condiciones. De esta manera, los usuarios pueden explorar la información de manera dinámica y obtener indicadores relevantes para la gestión de mantenimiento. En este contexto, la integración del modelo BIM con las herramientas de análisis de Power BI permitió transformar la información técnica del proyecto en paneles de control interactivos, los cuales facilitaron la interpretación de los datos, la identificación de elementos críticos y el apoyo a la toma de decisiones relacionadas con la planificación y gestión del mantenimiento de la infraestructura educativa

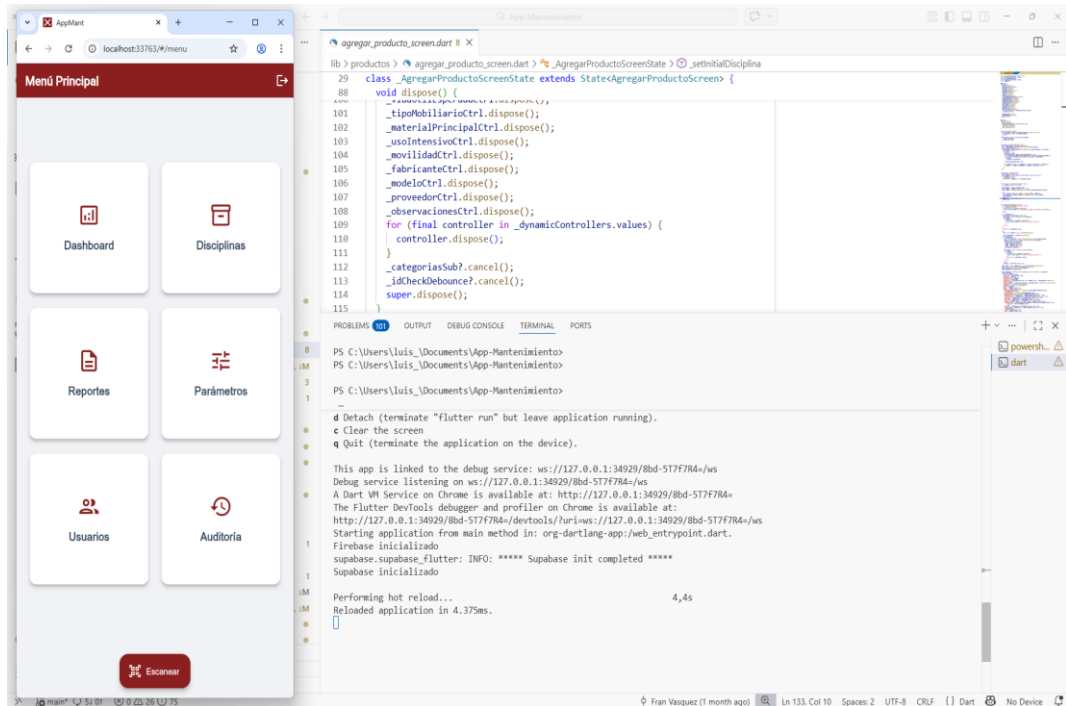
2.3.5. Análisis de datos

Con el aplicativo se recolectó información básica de los componentes arquitectónicos del IE (dimensiones, material, condiciones, edad entre otros) como inodoro, ducha, lavadero, dispensadores, pisos, paredes, puertas, mamparas, ventanas, luminarias, pintura entre otros elementos referidos a la arquitectura. Todos estos datos sirvieron para alimentar el aplicativo y que puede ser actualizado cuando sea necesario. El trabajo en conjunto fue la alimentación de información mediante el aplicativo al BIM. El Software BIM (Autodesk Revit, Power BI), que permitió la creación de modelos tridimensionales detallados y la simulación de intervenciones de mantenimiento, con reportes sobre el uso de recursos.

III. RESULTADOS

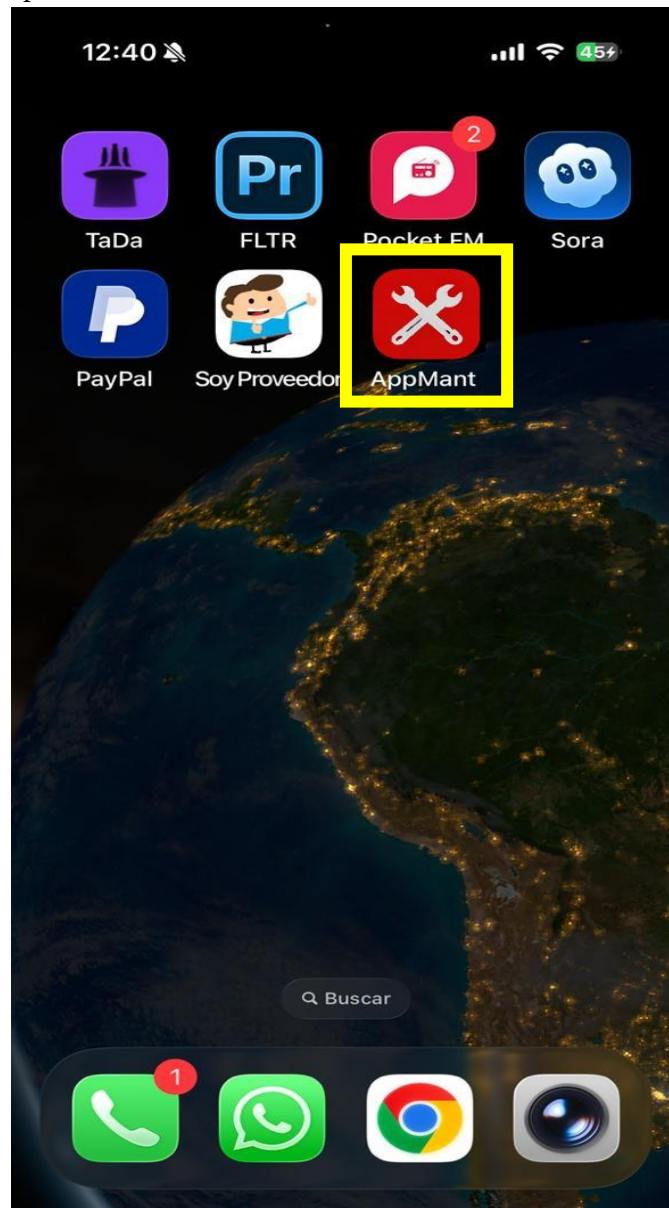
3.1. Diseñar y desarrollar un aplicativo para la recolección de información para la gestión del mantenimiento en la Institución Educativa Ramón Castilla y Marquesado.

El desarrollo del aplicativo App-Mantenimiento permite implementar una solución tecnológica integral orientada a la gestión digital de activos dentro de un entorno BIM interoperable. La integración de tecnologías como Flutter, Firebase, Supabase, Dynamo, Revit y Power BI permite consolidar un ecosistema tecnológico capaz de gestionar activos provenientes del modelo BIM, registrar incidencias de mantenimiento, actualizar parámetros en tiempo real y mantener un historial de información asociado a cada elemento. Asimismo, la incorporación de herramientas de visualización analítica permite representar la información mediante dashboards, facilitando el análisis y la toma de decisiones. La arquitectura modular adoptada en el desarrollo del sistema garantizando escalabilidad, mantenibilidad y una adecuada separación de responsabilidades entre los diferentes componentes del aplicativo. Esto permite que la plataforma pueda evolucionar hacia nuevos módulos o ampliaciones en otras disciplinas sin afectar la estructura principal del sistema. Desde el punto de vista técnico, el aplicativo evidencia la implementación de mecanismos de autenticación segura, el uso de bases de datos NoSQL con sincronización en tiempo real, la interoperabilidad bidireccional con el modelo BIM, así como una gestión estructurada de activos que incluye control de usuarios, auditoría de actividades y despliegue multiplataforma. En consecuencia, el proyecto no solo cumple con los objetivos planteados, sino que también demuestra la viabilidad técnica de integrar herramientas BIM con aplicaciones móviles, contribuyendo a optimizar los procesos de gestión y mantenimiento de infraestructura educativa mediante el uso de tecnologías digitales interoperables.

Figura 123*Proceso de compilación y Hot Reload.*

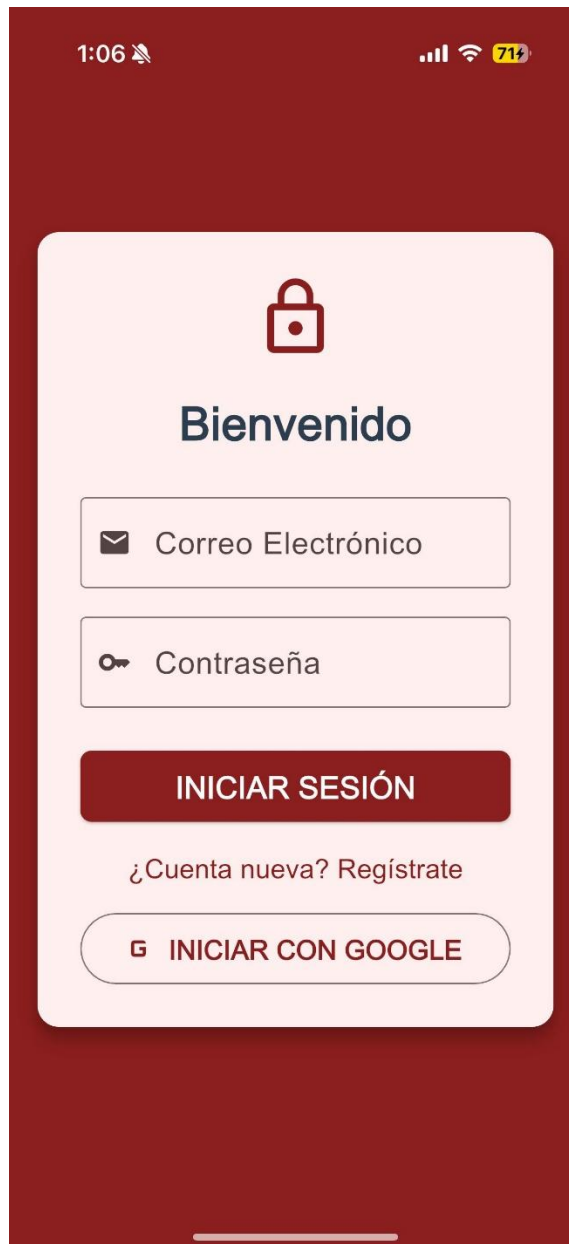
Nota. Datos tomados de Aplicativo (2026).

La figura muestra, el desarrollo del aplicativo final del aplicativo, donde se utilizó el modo debug de Flutter, el cual permitió ejecutar la aplicación y visualizar su funcionamiento en tiempo real. Este modo facilitó la detección y corrección de errores durante el proceso de programación. Asimismo, se empleó la funcionalidad Hot Reload, la cual permitió aplicar cambios en el código fuente sin necesidad de reiniciar completamente la aplicación. Para ello, se utilizó el comando `r` dentro de la consola de Flutter, lo que permitió recargar la interfaz y reflejar inmediatamente las modificaciones realizadas en el código. Este mecanismo optimizó el proceso de desarrollo al reducir los tiempos de compilación y permitir una retroalimentación inmediata sobre los cambios implementados. Además, la ejecución del aplicativo en entorno web mediante el navegador Chrome evidenció la capacidad multiplataforma de Flutter, permitiendo desplegar el sistema en diferentes plataformas, como Web y Android, utilizando una única base de código.

Figura 124*Aplicativo*

Nota. Datos tomados de Aplicativo (2026).

La figura muestra, el aplicativo instalado en celular, de versión Android, para su posterior uso.

Figura 125*Pantalla de Inicio de Sesión*

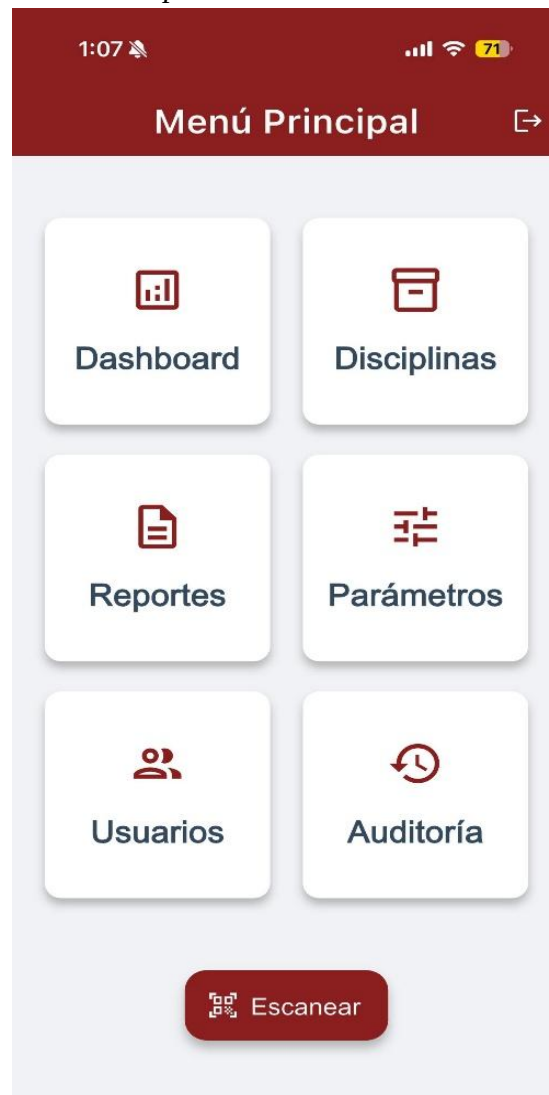
Nota. Datos tomados de Aplicativo (2026).

La figura muestra, la pantalla de inicio de session, que constituyó el punto de acceso principal al aplicativo. En esta interfaz se presentaron los campos destinados al ingreso del correo electrónico y la contraseña, los cuales fueron utilizados por el usuario para autenticarse dentro del sistema. Una vez ingresadas las credenciales, el usuario pudo iniciar el proceso de autenticación mediante el botón “Iniciar sesión”, el cual validó la información proporcionada a través del servicio Firebase Authentication. Asimismo, la interfaz incluyó una opción alternativa para iniciar sesión mediante una cuenta de Google, facilitando el

acceso al sistema. De esta manera, esta pantalla permitió implementar un mecanismo de autenticación segura, garantizando la validación de credenciales y el control de acceso al sistema, asegurando que únicamente los usuarios autorizados pudieran ingresar y utilizar las funcionalidades del aplicativo.

Figura 126

Menú Principal / Dashboard



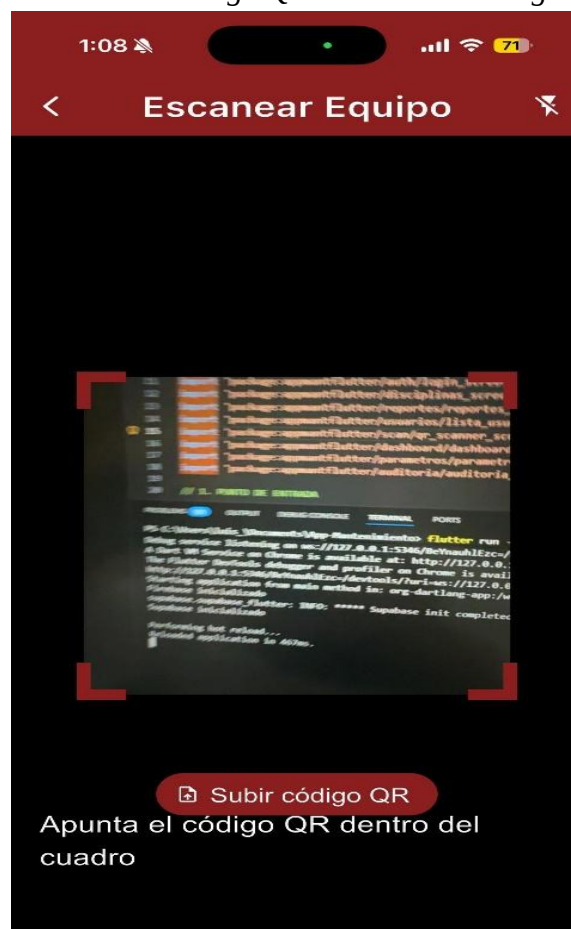
Nota. Datos tomados de Aplicativo (2026).

La figura muestra, el menú principal del sistema permitió acceder a los diferentes módulos funcionales del aplicativo, los cuales se presentaron mediante tarjetas de navegación organizadas de forma visual e intuitiva. En esta interfaz se mostraron las opciones correspondientes a Dashboard, Disciplinas, Reportes, Parámetros, Usuarios y Auditoría, facilitando el acceso a las principales funcionalidades del sistema.

Cada módulo fue representado mediante una tarjeta con iconografía descriptiva, lo cual permitió una identificación rápida de las funciones disponibles y una navegación sencilla dentro del aplicativo. Asimismo, el sistema incluyó un botón de escaneo, destinado a activar la funcionalidad de lectura de códigos QR para la identificación rápida de activos. De esta manera, el panel principal permitió una organización estructurada de los módulos del sistema, facilitando la visualización de las funcionalidades operativas y proporcionando una interfaz clara e intuitiva para la navegación del usuario.

Figura 127

Escáner de Código QR Escáner de Código QR



Nota. Datos tomados de Aplicativo (2026).

El módulo de escaneo permite identificar activos de manera rápida mediante la lectura de códigos QR asociados a cada elemento físico registrado en el sistema. En esta interfaz se activó la cámara del dispositivo móvil, la cual permitió capturar y procesar el código QR ubicado dentro del área de escaneo mostrada en pantalla. Una vez detectado el código, el sistema extrae automáticamente el identificador del activo y realiza una consulta

en tiempo real a la base de datos, con el fin de recuperar la información correspondiente al equipo registrado. Posteriormente, el aplicativo muestra los datos del activo identificado, permitiendo al usuario acceder a su información detallada. De esta manera, esta funcionalidad evidencia la integración del sistema con el hardware del dispositivo, específicamente con la cámara, así como la capacidad del aplicativo para realizar consultas dinámicas a la base de datos, facilitando la identificación y gestión de activos en campo.

Figura 128

Visualización y gestión de activos registrados.



Nota. Datos tomados de Aplicativo (2026).

La figura muestra, el módulo de productos que permite visualizar y administrar los activos registrados en el sistema, los cuales fueron importados previamente desde el modelo BIM. En esta pantalla se presentó un listado de dispositivos de iluminación, donde

los activos fueron organizados según su estado operativo, diferenciando aquellos que se encontraban operativos de los que estaban fuera de servicio. Cada activo fue mostrado mediante una tarjeta informativa que incluyó el nombre del equipo, su estado de funcionamiento y un espacio destinado para la imagen asociada al activo, en caso de que esta estuviera disponible. Asimismo, el sistema permitió acceder al detalle del activo, donde se pudo consultar información más específica relacionada con sus características y parámetros técnicos. De esta manera, este módulo evidenció la correcta implementación de las operaciones CRUD para la gestión de activos, así como la sincronización de los parámetros provenientes del modelo BIM desarrollado en Revit y el almacenamiento estructurado de la información en la base de datos Firestore, lo cual permitió gestionar de forma eficiente los activos dentro del sistema.

Figura 129

Registro y consulta de reportes de mantenimiento.

Nota. Datos tomados de Aplicativo (2026).

El módulo de reportes permite registrar incidencias relacionadas con los activos del sistema, facilitando la gestión y seguimiento de las actividades de mantenimiento. En esta interfaz se presenta un formulario donde el usuario pudo seleccionar el tipo de reporte, el activo asociado, así como visualizar información relacionada con su ubicación, categoría y estado actual. Asimismo, el sistema permite registrar la fecha de inspección y el estado detectado durante la revisión del equipo, lo que facilita documentar las condiciones en las que se encontraba el activo en el momento de la inspección. Esta información es almacenada en la base de datos para su posterior consulta y análisis. De esta manera, este módulo permite actualizar dinámicamente el estado de los activos, mantener un historial de reportes de mantenimiento y garantizar la trazabilidad de las intervenciones realizadas, contribuyendo a una gestión operativa más eficiente del mantenimiento de los equipos.

3.2. Implementar y evaluar el aplicativo en la recolección de datos de infraestructura para la gestión y mantenimiento en la Institución Educativa Ramón Castilla y Marquesado.

Figura 130

Listado de reportes generados y actualización de parámetros para el modelo BIM



Nota. Datos tomados de Aplicativo (2026).

La figura muestra la interfaz del aplicativo correspondiente al listado de reportes generados durante el proceso de inspección de los activos de infraestructura. En la pantalla se visualizan los registros almacenados en el sistema, donde cada reporte incluyó información como el número de registro, la fecha y hora de la inspección, el tipo de reporte y el estado operativo del activo evaluado. Estos reportes fueron generados a partir de las inspecciones realizadas en campo y permitieron registrar información técnica relevante

sobre las condiciones de los elementos evaluados. A partir de estos registros, el sistema actualiza parámetros asociados al activo, tales como Estado Operativo, Condición Física, Nivel de Criticidad, Riesgo Normativo, Acción Recomendada, Costo Estimado y Fecha de Última Inspección. La información almacenada en el aplicativo constituyó la base de datos de mantenimiento de los activos inspeccionados, permitiendo organizar y sistematizar los resultados obtenidos durante el levantamiento en campo. Asimismo, estos parámetros registrados son posteriormente exportados al modelo BIM desarrollado en Revit, lo que permite integrar la información técnica de mantenimiento con el modelo digital de la infraestructura.

Figura 131

Listado de reportes generados para “Arquitectura – Base”



ARQUITECTURA - BASE	
tarrajeo derrames e=1.5 cm	
tarrajeo derrames e=1.5 cm	
tarrajeo derrames e=1.5 cm	
tarrajeo derrames e=1.5 cm	
tarrajeo derrames e=1.5 cm	
tarrajeo derrames e=1.5 cm	
tarrajeo derrames e=1.5 cm	
tarrajeo derrames e=1.5 cm	
tarrajeo derrames e=1.5 cm	
tarrajeo derrames e=1.5 cm	
tarrajeo derrames e=1.5 cm	
tarrajeo derrames e=1.5 cm	
tarrajeo derrames e=1.5 cm	
tarrajeo derrames e=1.5 cm	
Generar Excel	

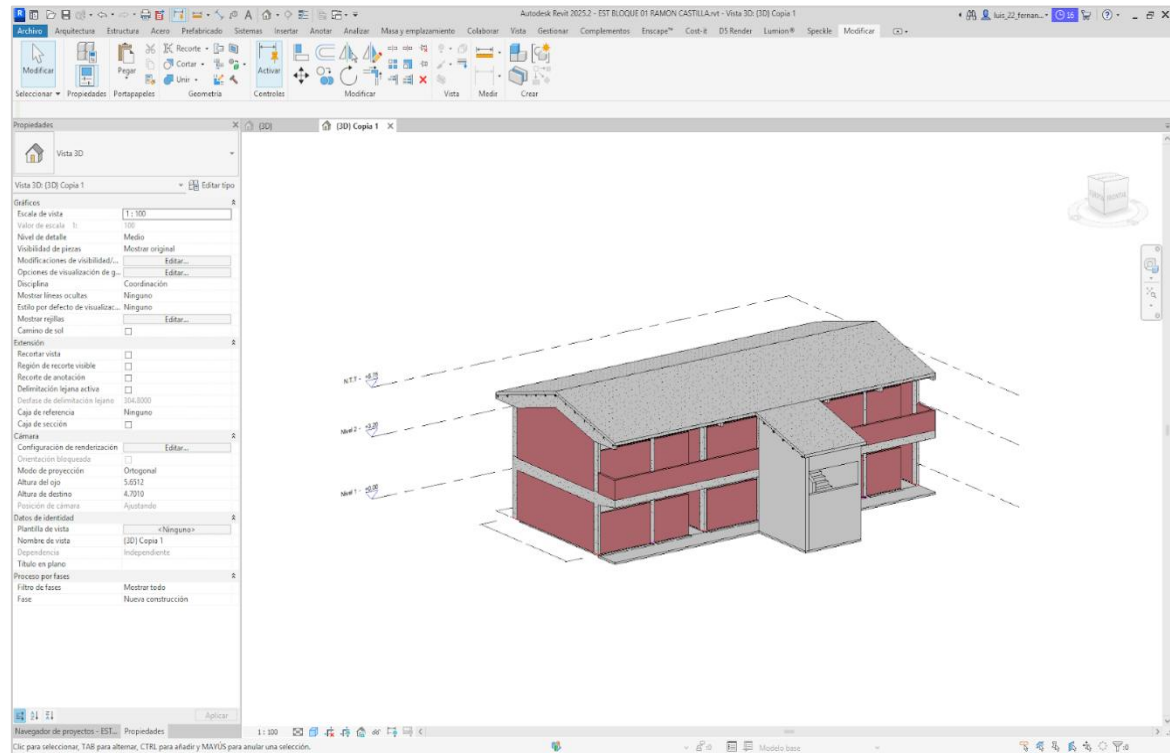
Nota. Datos tomados de Aplicativo (2026).

La figura muestra la interfaz del aplicativo móvil correspondiente al módulo “Arquitectura – Base”, donde se visualiza el listado de activos registrados dentro del sistema, en este caso elementos constructivos asociados al tarrajeo de derrames con un espesor de 1.5 cm. En esta sección del aplicativo se consolida la información actualizada de los activos evaluados durante el proceso de inspección y levantamiento en campo. Asimismo, en la parte inferior de la interfaz se observa la opción “Generar Excel”, la cual permitió exportar la base de datos registrada en el aplicativo hacia un archivo en formato Excel. Este archivo contiene los parámetros actualizados de los activos, incluyendo información relacionada con su estado operativo, condición física, nivel de criticidad, riesgo normativo, acciones recomendadas, costos estimados y fecha de última inspección. La exportación de estos datos constituyó una etapa fundamental dentro del flujo de trabajo implementado, ya que el archivo generado es posteriormente utilizado para su procesamiento e importación al modelo BIM mediante el uso de Dynamo en Revit. A través de este procedimiento, los parámetros de mantenimiento registrados en campo se integran al modelo digital de la infraestructura, permitiendo actualizar automáticamente la información asociada a los elementos del modelo.

3.3. Integrar la información recopilada del aplicativo mediante el modelo BIM para optimizar la gestión del mantenimiento en la Institución Educativa Ramón Castilla y Marquesado.

Figura 132

Modelado final de estructuras del bloque I de la IE



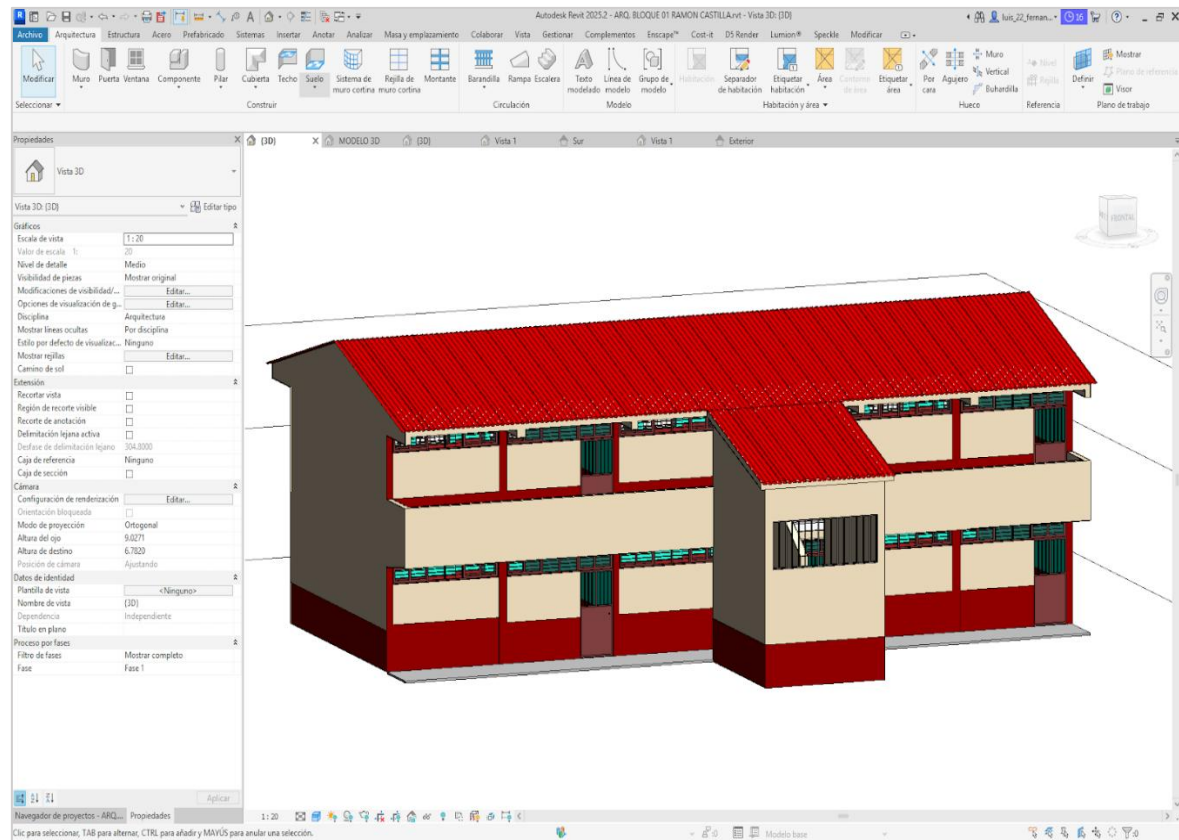
Nota. Datos tomados de Revit 2025.

La figura muestra el modelado final de la estructura de la Institución Educativa Ramón Castilla y Marquesado desarrollado en Revit 2025. El modelo tridimensional integra los principales elementos estructurales, tales como muros, losas y cubierta, constituyendo la base digital para la incorporación de la información recopilada mediante el aplicativo desarrollado. Este modelo BIM permitió asociar parámetros técnicos y datos de mantenimiento a cada componente, facilitando una gestión estructurada y eficiente de la infraestructura educativa.

12

Figura 133

Modelado final de arquitectura del bloque I de la IE

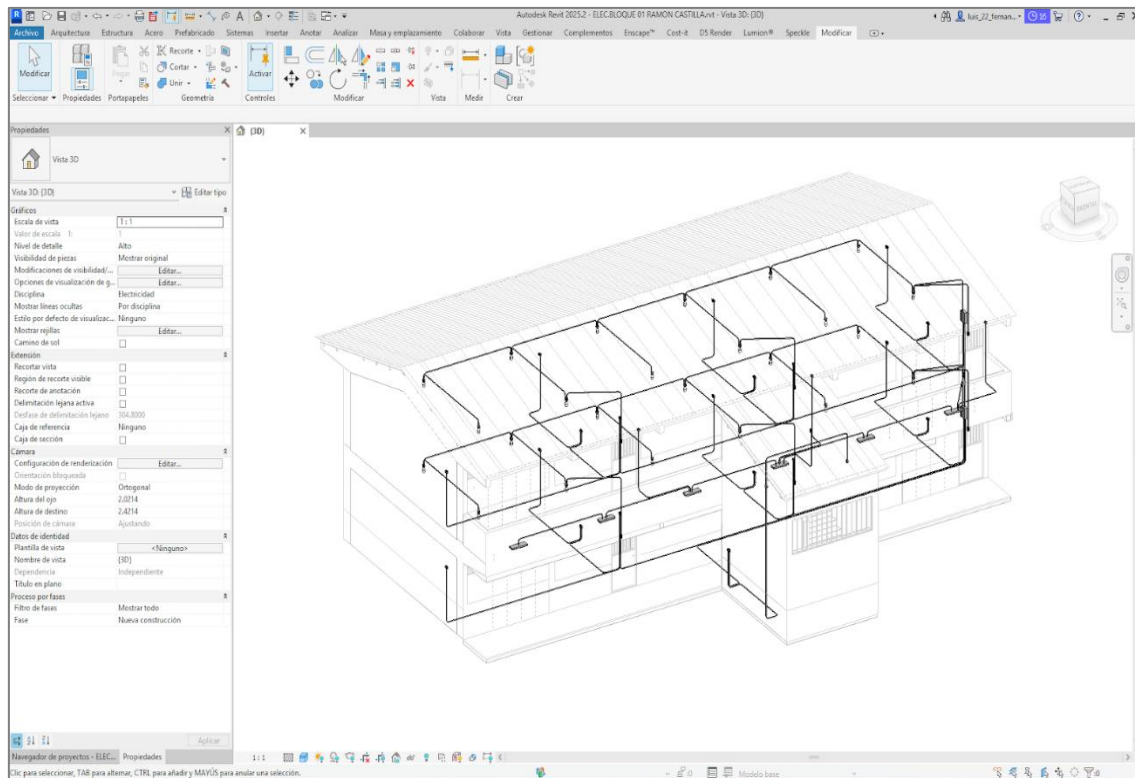


Nota. Datos tomados de Revit 2025.

La figura muestra el modelado final arquitectónico del bloque I de la Institución Educativa Ramón Castilla y Marquesado desarrollado en Revit 2025. En el modelo se incorporaron acabados como tarrajeo en muros y losas, pintura, contrapisos, cobertura final con cielorraso, así como la estandarización de elementos de carpintería metálica, incluyendo una familia única de puerta metálica de 2.10 m x 1.00 m y ventanas de distintas dimensiones con rejas metálicas. Este modelado permitió integrar la información recopilada mediante el aplicativo, asociando parámetros de mantenimiento a cada componente arquitectónico para optimizar la gestión de la infraestructura.

Figura 134

Modelado final de eléctricas del bloque I de la IE

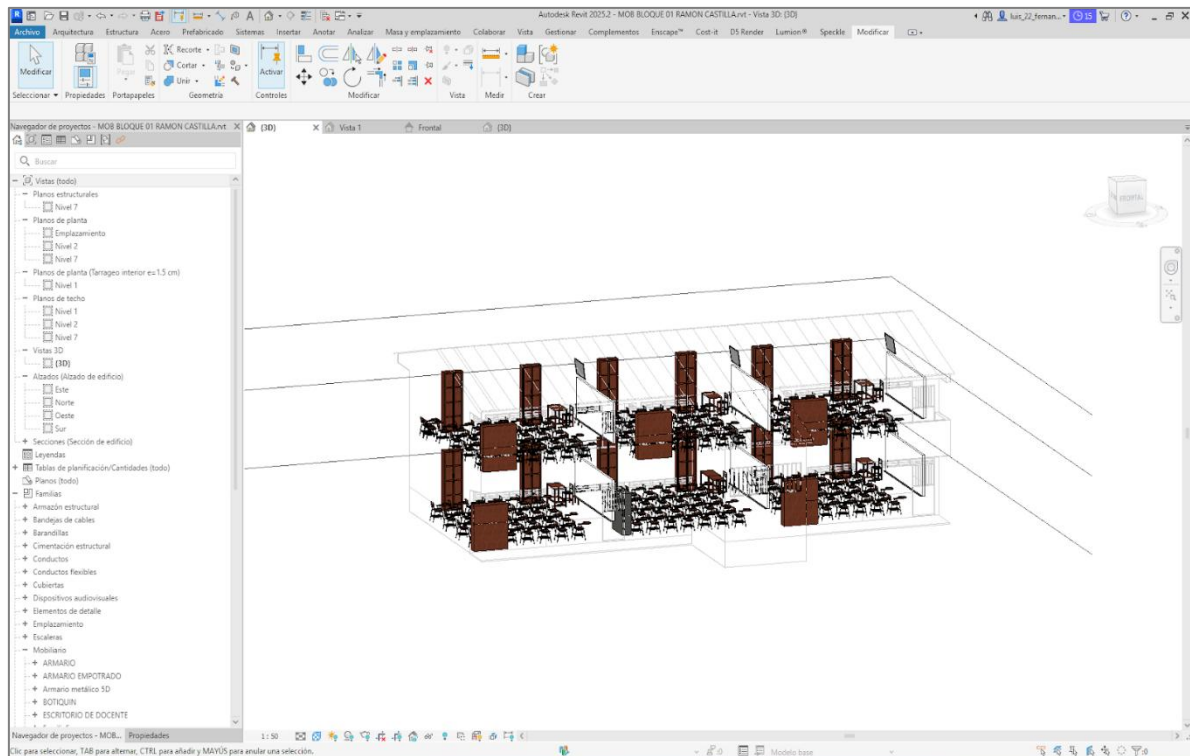


Nota. Datos tomados de Revit 2025.

La figura muestra el modelado final de las instalaciones eléctricas del Bloque I de la Institución Educativa Ramón Castilla y Marquesado, desarrollado en Revit 2025. El modelo integra la distribución de canalizaciones, circuitos eléctricos, tablero de distribución empotrado, tomacorrientes con dimensiones paramétricas de 110 mm x 60 mm y distintos tipos de luminarias, incluyendo luminarias LED tipo bombilla en aulas, paneles LED empotrados en el pasillo del primer nivel y downlights empotrados en el segundo nivel y la escalera. La integración de estos elementos en el entorno BIM permitió asociar información recopilada mediante el aplicativo, facilitando la identificación de componentes, la planificación de intervenciones y la optimización de la gestión del mantenimiento eléctrico.

Figura 135

Modelado final de mobiliario del bloque I de la IE



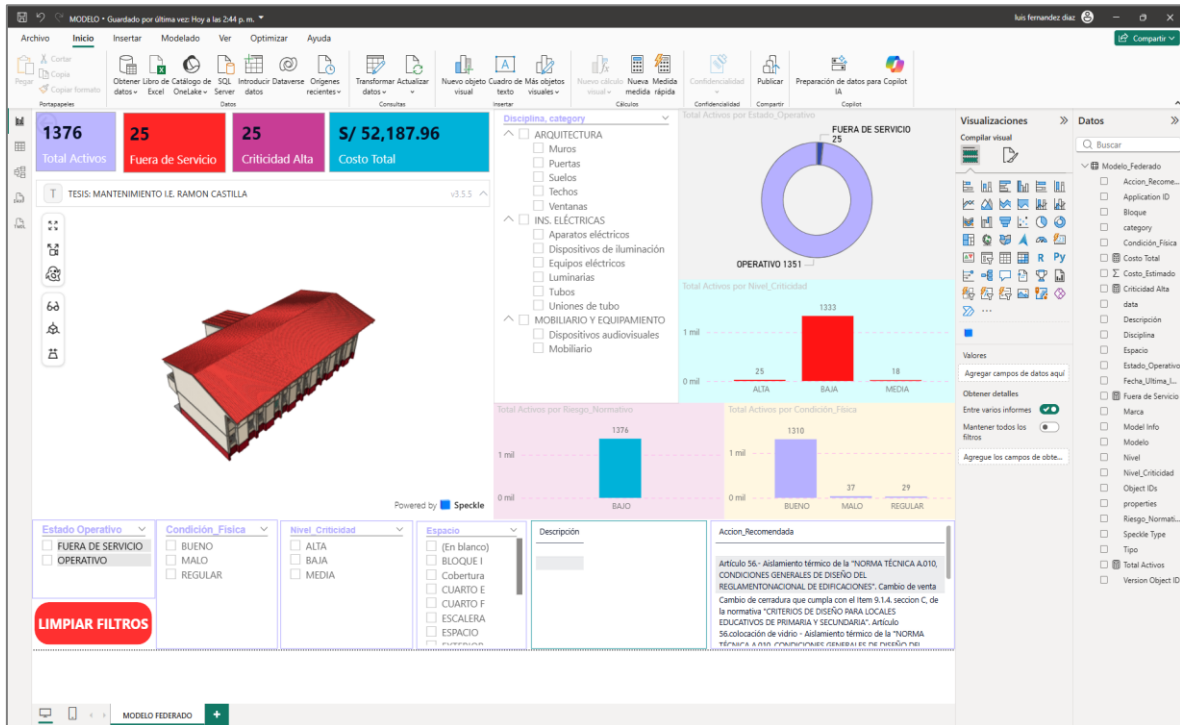
Nota. Datos tomados de Revit 2025.

La figura muestra el modelado final de mobiliario del bloque I de la Institución Educativa Ramón Castilla y Marquesado desarrollado en Revit 2025. En el modelo se incorporaron la distribución de sillas unipersonales organizadas en filas, el escritorio y silla del docente en la parte frontal, pizarra acrílica y soporte para TV (1.13×0.65 m). Asimismo, se incluyeron dos armarios de madera con dimensiones de $2.45 \times 0.85 \times 0.30$ m y $1.85 \times 1.50 \times 0.40$ m. El modelo permite visualizar la organización espacial y funcional de las aulas del bloque.

Al tener la información con los filtros tanto, por estado operativo, condición física, nivel de criticidad, espacio, así como de las disciplinas trabajadas se obtiene lo siguiente,

Figura 136

Panel de control para la gestión y monitoreo de activos de infraestructura

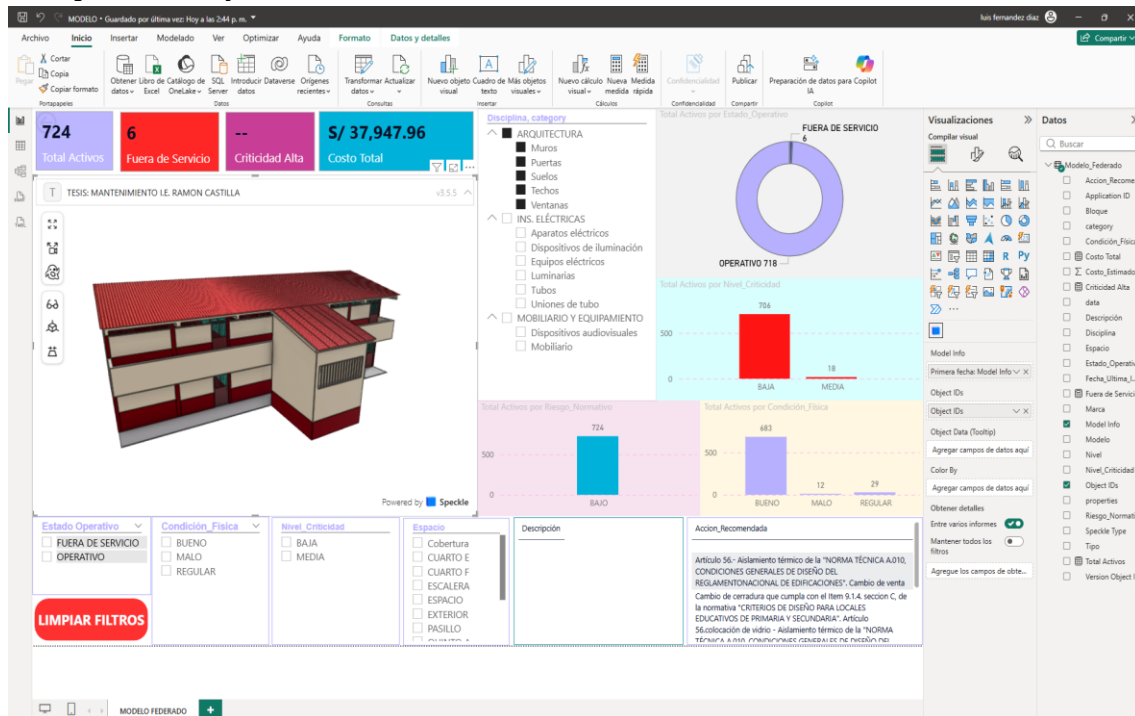


Nota. Datos tomados de Power bi (2026).

La figura muestra el resumen de los datos observados, donde se registraron 1376 activos totales en el sistema; de estos, 25 activos están fuera de servicio, mientras que 1351 se encuentran operativos, además existen 25 activos con criticidad alta. El costo total estimado asociado es de S/ 52,187.96. La condición física de los activos, 1310 activos son buenos, 37 activos malos y 29 regular. Y referente al nivel de criticidad, Alta son 25 y bajo 18. Finalmente, de todos los 1376 activos aparecen clasificados con riesgo bajo. El panel también incluye filtros de análisis como: estado operativo, condición física, nivel de criticidad, espacio o ambiente (bloques, cuartos, escaleras, etc.), permitiendo segmentar los activos para evaluar su estado y priorizar acciones de mantenimiento.

Figura 137

Disciplina de arquitectura

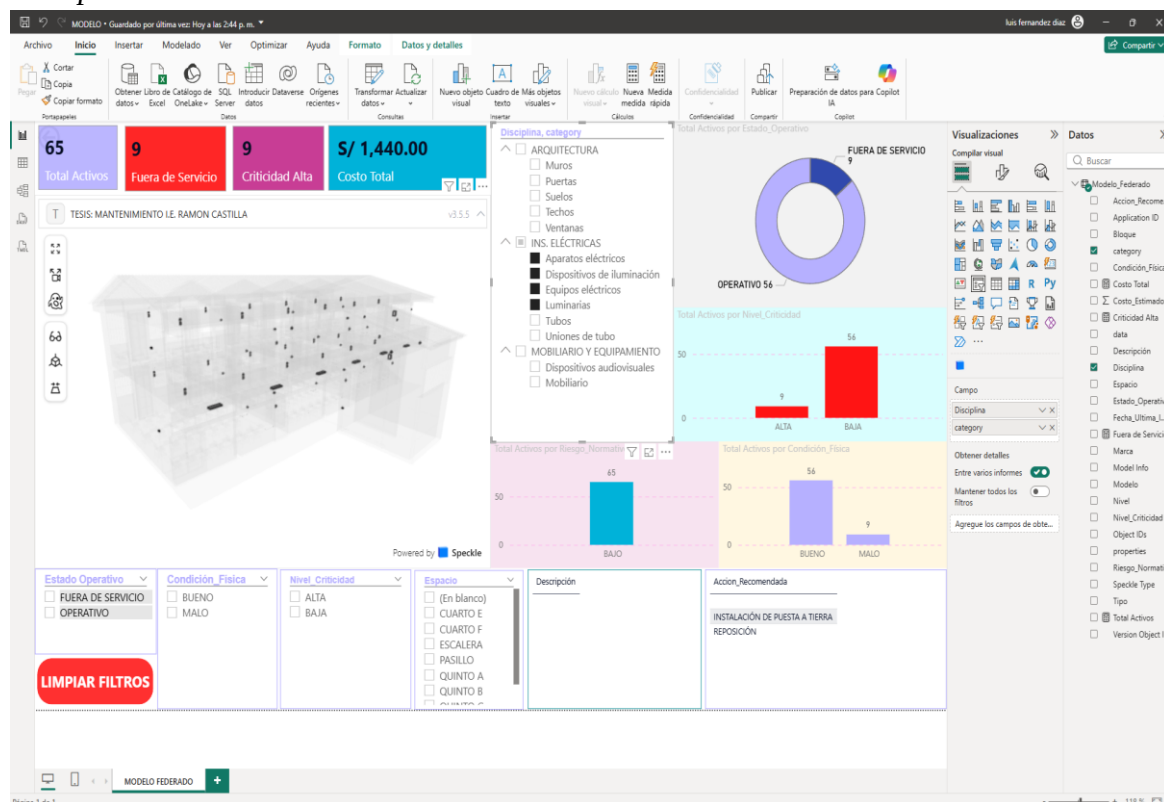


Nota. Datos tomados de Power bi (2026).

La figura muestra el resumen de los datos observados, para arquitectura, donde se registran 724 activos arquitectónicos en total; donde de estos, 718 activos se encuentran operativos y 6 activos están fuera de servicio. El costo total estimado de intervención o mantenimiento es de S/ 37,947.96. No se reportan activos con criticidad alta en el filtro aplicado, solo siendo Baja 706 activos y media 18 activos. Los 724 activos se clasifican con riesgo normativo bajo, cuya condición física es de (Bueno: 683 activos/ Malo: 12 activos /Regular: 29 activos). El panel también permite analizar los activos mediante diferentes filtros, como estado operativo, condición física, nivel de criticidad y espacio dentro de la infraestructura (cobertura, cuartos, escaleras, pasillos, entre otros). Asimismo, se visualiza un modelo 3D del edificio educativo, que facilita la identificación espacial de los elementos evaluados para la gestión y priorización de actividades de mantenimiento.

Figura 138

Disciplina de eléctricas

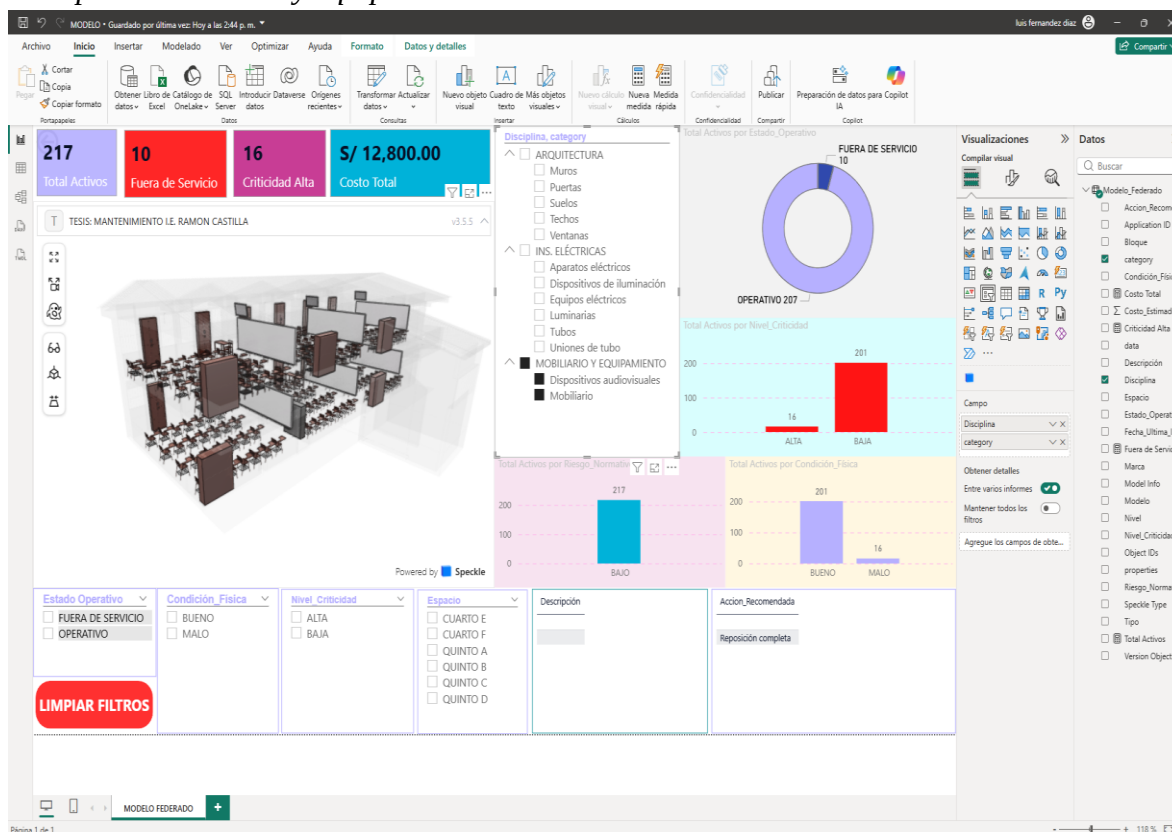


Nota. Datos tomados de Power bi (2026).

La figura muestra el resumen de los datos observados, para eléctricas, donde se registran 65 activos eléctricos en total; de estos, 56 activos se encuentran operativos y 9 activos están fuera de servicio. El costo total estimado de intervención o mantenimiento es de S/ 1,440.00. El Nivel de criticidad de los activos (Alta: 9 activos / Baja: 56 activos). Los 65 activos se clasifican con riesgo normativo bajo, cuya Condición física de los activos, son 56 activos buenos y 9 malos

Figura 139

Disciplina mobiliario y equipamientos

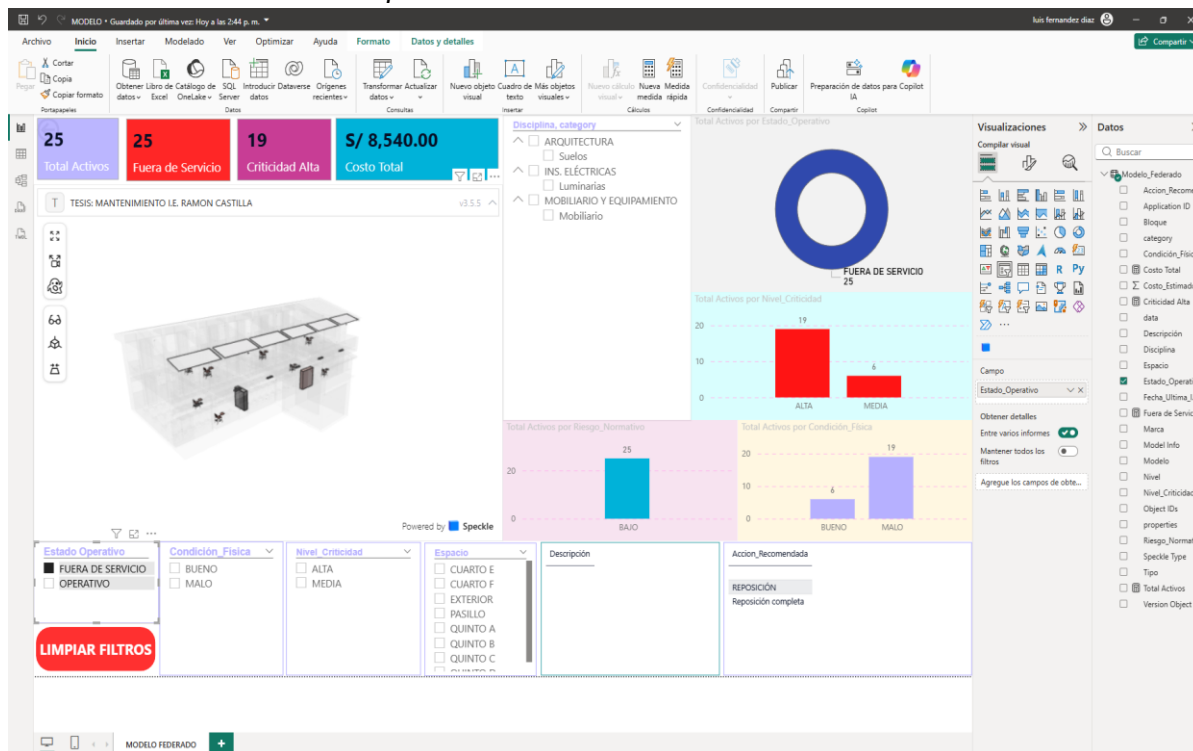


Nota. Datos tomados de Power bi (2026).

La figura muestra el resumen de los datos observados, para mobiliario y equipamientos, donde se registran 217 activos de mobiliario y equipamiento en total; de estos, 207 activos se encuentran operativos y 10 activos están fuera de servicio. El costo total estimado de intervención o mantenimiento es de S/ 12,800.00. Nivel de criticidad de los activos (Alta: 16 activos / Baja: 201 activos). Los 217 activos se clasifican con riesgo normativo bajo, cuya condición física de los activos, son 201 activos buenos y Malo 16 activos.

Figura 140

Análisis de activos en estado fuera de servicio



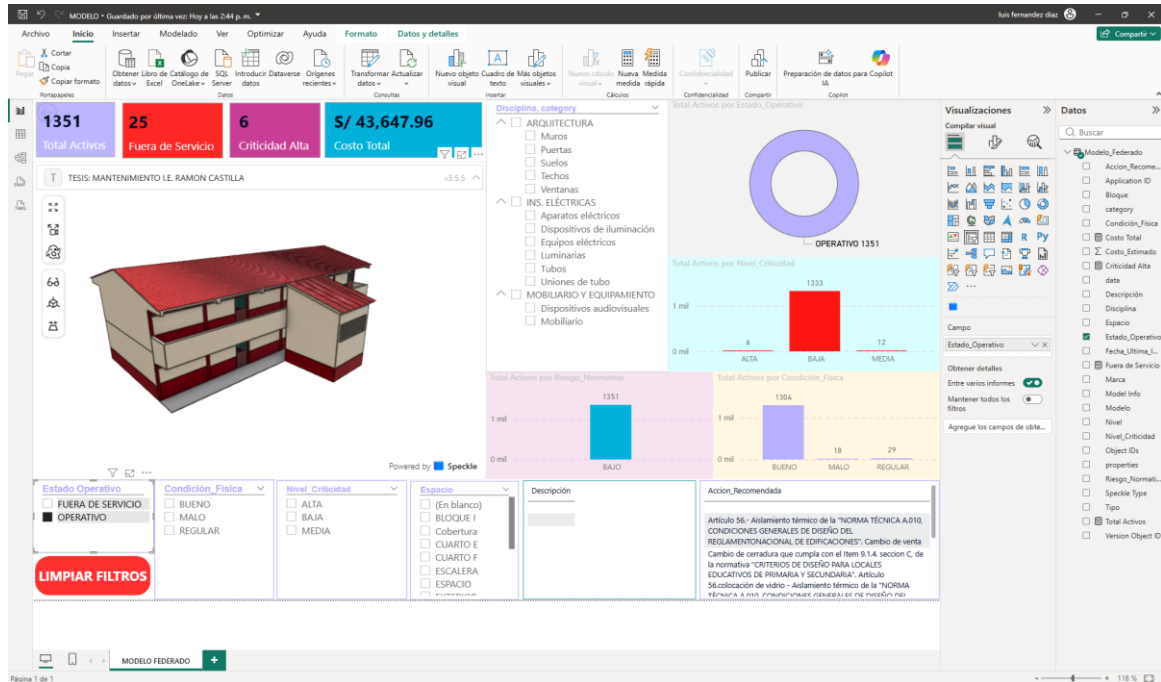
Nota. Datos tomados de Power bi (2026).

La figura presenta el análisis de los activos filtrados bajo la condición estado operativo. Este análisis permitió identificar los elementos de la infraestructura que actualmente no se encuentran operativos y que requieren algún tipo de intervención o mantenimiento. De acuerdo con los datos observados en el panel, se registran 25 activos en condición de fuera de servicio, lo que indica que la totalidad de los elementos considerados en este filtro presentan algún tipo de falla o deterioro que impide su funcionamiento adecuado. En consecuencia, no se registran activos operativos dentro del conjunto de datos analizado. En relación con el nivel de criticidad, se observa que 19 activos presentan criticidad alta, mientras que 6 activos presentan criticidad media, lo que evidencia que una gran proporción de los elementos fuera de servicio requiere atención prioritaria debido a su impacto en el funcionamiento de la infraestructura. Respecto al riesgo normativo, todos los 25 activos se clasifican con riesgo normativo bajo, lo que indica que, aunque presentan fallas operativas, estas no representan un incumplimiento crítico de las normativas vigentes o un riesgo inmediato para la seguridad. En cuanto a la condición física de los activos, se identifican 19 activos en estado físico malo y 6 activos en estado bueno. Esto sugiere que la mayoría de los elementos fuera de servicio presentan deterioro significativo, lo que explica

su condición operativa actual. Finalmente, el análisis muestra que el costo total estimado de intervención asciende a S/ 8,540.00, lo cual representa el monto requerido para ejecutar las acciones de mantenimiento o reposición necesarias para restablecer el funcionamiento de los activos evaluado

Figura 141

Resumen de los datos observados



Nota. Datos tomados de Power bi (2026).

La figura presenta los activos filtrados bajo la condición estado operativo: operativo, lo que permite evaluar el comportamiento y las características de los elementos que actualmente se encuentran funcionando dentro de la infraestructura de la institución educativa. De acuerdo con los resultados obtenidos, se registraron 1351 activos operativos dentro del sistema. Esto indica que la mayor parte de los elementos evaluados se encuentran funcionando adecuadamente y cumplen con su función dentro de la infraestructura. Los activos que presentaron condición de fuera de servicio han sido excluidos del filtro aplicado, permitiendo analizar únicamente aquellos que se mantienen en funcionamiento. En relación con el nivel de criticidad, se observa que la gran mayoría de los activos presentan criticidad baja (1333 activos), lo que sugiere que su posible falla tendría un impacto reducido en el funcionamiento general de la infraestructura. Sin embargo, se identifican 12 activos con criticidad media y 6 activos con criticidad alta, los cuales requieren mayor atención dentro

de las actividades de inspección y mantenimiento, ya que su deterioro podría afectar de manera más significativa el desempeño del sistema. Respecto al riesgo normativo, todos los 1351 activos operativos presentan un nivel de riesgo bajo, lo que indica que estos elementos cumplen con las condiciones normativas básicas y no representan un riesgo significativo en términos de seguridad o cumplimiento técnico. En cuanto a la condición física de los activos, se observa que 1304 activos se encuentran en buen estado, lo que confirma el adecuado estado general de la infraestructura. No obstante, se identifican 18 activos en condición física mala, los cuales, aunque aún se encuentran operativos, presentan deterioro que podría derivar en futuras fallas si no se realizan acciones de mantenimiento oportunas. Finalmente, el análisis muestra un costo total estimado de intervención de S/ 43,647.96, el cual corresponde a las acciones de mantenimiento preventivo o correctivo necesarias para mantener en condiciones adecuadas los activos operativos identificados.

IV. DISCUSIÓN

Del diseño y desarrollo de aplicativo para la recolección de información para la gestión del mantenimiento en la Institución Educativa Ramón Castilla y Marquesado. Se implementó una solución tecnológica orientada a la gestión digital de activos dentro de un entorno BIM interoperable, integrando diversas herramientas tecnológicas como Flutter, Firebase, Supabase, Dynamo, Revit y Power BI. Esta integración tecnológica permitió consolidar un ecosistema digital capaz de registrar información de los activos, gestionar incidencias de mantenimiento y mantener un historial actualizado de las intervenciones realizadas en cada elemento de la infraestructura. El desarrollo del aplicativo demuestra la viabilidad de integrar herramientas de modelado BIM con plataformas móviles para la gestión de mantenimiento. La utilización del framework Flutter permitió desarrollar una aplicación multiplataforma, capaz de ejecutarse tanto en dispositivos Android como en entornos web, optimizando los procesos de desarrollo mediante funcionalidades como el modo debug y el Hot Reload, que facilitaron la detección y corrección de errores durante la programación. Este proceso permitió mejorar la eficiencia del desarrollo y garantizar la estabilidad funcional del sistema implementado. Asimismo, la implementación de un sistema de autenticación mediante Firebase Authentication permitió garantizar el control de acceso al aplicativo, asegurando que únicamente los usuarios autorizados puedan ingresar al sistema y gestionar la información registrada. Este mecanismo fortalece la seguridad de la plataforma y permite establecer un control adecuado sobre los usuarios que interactúan con los activos registrados en el sistema. En relación con la estructura funcional del aplicativo, se identificó la implementación de un menú principal organizado mediante módulos operativos como Dashboard, Disciplinas, Reportes, Parámetros, Usuarios y Auditoría. Esta estructura modular facilita la navegación del usuario y permite una gestión organizada de las diferentes funcionalidades del sistema, lo que contribuye a mejorar la eficiencia en el manejo de la información relacionada con los activos de la infraestructura educativa.

Por otro lado, el sistema incorpora un módulo de escaneo de códigos QR que permite identificar de manera rápida los activos físicos registrados en la base de datos. Mediante la activación de la cámara del dispositivo móvil, el aplicativo puede capturar el código QR asociado a cada elemento y consultar automáticamente la información correspondiente almacenada en la base de datos. Esta funcionalidad mejora significativamente los procesos de identificación y seguimiento de los equipos en campo,

reduciendo tiempos de registro y minimizando posibles errores en la gestión de los activos. De igual manera, el aplicativo permite visualizar y administrar los activos registrados en el sistema mediante un módulo que organiza los equipos según su estado operativo. En esta interfaz se muestran los activos importados desde el modelo BIM desarrollado en Revit, permitiendo consultar información técnica detallada y evidenciando la correcta implementación de operaciones CRUD (crear, leer, actualizar y eliminar) para la gestión de activos dentro del sistema. La sincronización entre el modelo BIM y la base de datos Firestore permite mantener la información actualizada y estructurada, lo que facilita la administración de los elementos de infraestructura. Finalmente, el sistema incorpora un módulo de registro de reportes de mantenimiento que permite documentar incidencias detectadas durante las inspecciones realizadas a los activos. Este módulo permite registrar información como el tipo de reporte, el activo asociado, su ubicación, categoría, estado actual, fecha de inspección y estado detectado. La información registrada es almacenada en la base de datos, lo que permite mantener un historial de mantenimiento y garantizar la trazabilidad de las intervenciones realizadas. Esta funcionalidad contribuye a mejorar la planificación de las actividades de mantenimiento y fortalece la gestión operativa de la infraestructura. En conjunto, los resultados obtenidos demuestran que el aplicativo desarrollado cumple con el objetivo propuesto, ya que permite implementar un sistema digital para la gestión del mantenimiento basado en tecnologías interoperables. La integración entre BIM, aplicaciones móviles y bases de datos en tiempo real representa una herramienta eficiente para la administración de activos, permitiendo mejorar la organización de la información, optimizar los procesos de mantenimiento y facilitar la toma de decisiones en la gestión de infraestructura educativa.

De la implementación y evaluación del aplicativo en la recolección de datos de infraestructura para la gestión y mantenimiento en la Institución Educativa Ramón Castilla y Marquesado. Los resultados obtenidos evidencian que el aplicativo desarrollado permitió registrar y sistematizar información técnica proveniente de las inspecciones realizadas en campo, generando reportes estructurados que facilitan el análisis y la gestión del mantenimiento de los activos de infraestructura. En la interfaz del sistema se visualiza el listado de reportes generados durante el proceso de inspección, donde cada registro contiene información relevante como el número de reporte, la fecha y hora de la inspección, el tipo de reporte y el estado operativo del activo evaluado. Esta información constituye un registro organizado de las condiciones observadas en los elementos inspeccionados, lo que

contribuye a mejorar el control y seguimiento de las actividades de mantenimiento. Asimismo, los reportes generados durante las inspecciones permitieron actualizar diversos parámetros asociados a cada activo, tales como el estado operativo, la condición física, el nivel de criticidad, el riesgo normativo, la acción recomendada, el costo estimado y la fecha de la última inspección. Estos parámetros permiten caracterizar de manera más precisa el estado de los elementos de la infraestructura, facilitando la priorización de intervenciones de mantenimiento y la toma de decisiones basada en información técnica actualizada. De esta manera, el aplicativo no solo cumple la función de registro de datos, sino que también permite estructurar una base de datos de mantenimiento que organiza y sistematiza los resultados obtenidos durante el proceso de levantamiento de información en campo. La información registrada en el aplicativo fue posteriormente exportada hacia un archivo en formato Excel, el cual constituye una etapa clave dentro del flujo de trabajo implementado. Este archivo contiene los parámetros actualizados de los activos inspeccionados y permitió realizar el procesamiento de la información para su integración con el modelo BIM desarrollado en Revit. A través del uso de Dynamo, los datos recopilados en campo pueden ser importados al modelo digital de la infraestructura, permitiendo actualizar automáticamente los parámetros asociados a cada elemento del modelo. Esta integración entre la base de datos generada en el aplicativo y el modelo BIM facilita la vinculación entre la información técnica de mantenimiento y la representación digital de la infraestructura. Asimismo, el aplicativo permitió consolidar la información de los activos evaluados dentro de módulos específicos, como el correspondiente a “Arquitectura”, “eléctricas” y “mobiliario y equipamientos”, donde se visualizan los elementos constructivos registrados en el sistema. En este módulo se centraliza la información actualizada de los activos inspeccionados, permitiendo visualizar de manera organizada los elementos evaluados durante el proceso de inspección. La posibilidad de generar automáticamente archivos Excel a partir de la información registrada facilita la gestión de los datos recopilados y permite su posterior utilización en procesos de análisis, modelado y actualización del modelo BIM. De forma concluyente, los resultados obtenidos demuestran que la implementación del aplicativo permitió optimizar el proceso de recolección de datos de infraestructura, ya que facilita el registro sistemático de información en campo, la actualización de parámetros técnicos de los activos y la integración de los datos con el modelo BIM de la infraestructura.

Estos hallazgos guardan relación con lo planteado por Zamora (2023), quien señala que la optimización de los procesos de operación y mantenimiento mediante BIM depende en gran medida de la adecuada estructuración de la información. No obstante, mientras Zamora propone una estructuración previa basada en listas de verificación (check list), el presente estudio avanza un paso más al automatizar la captura de estos parámetros directamente desde campo, reduciendo la dependencia de formatos manuales. Asimismo, el autor enfatiza la importancia de realizar recorridos en campo para el levantamiento de información precisa y su adecuada vinculación con el modelo BIM, señalando que este proceso debe apoyarse en herramientas digitales como tablets para mejorar la eficiencia. Este planteamiento es consistente con los resultados obtenidos, ya que el aplicativo empleado permite precisamente digitalizar este proceso de levantamiento, facilitando la recolección directa de datos y su posterior exportación a Excel para su integración automatizada mediante Dynamo. En contraste, mientras el estudio de Zamora describe un proceso aun parcialmente dependiente de la intervención manual y coordinación entre especialistas, la presente investigación evidencia una mayor integración tecnológica, reduciendo la fragmentación de la información.

Por otro lado, Abanto (2023) planteó que el proceso de gestión inicia con un levantamiento detallado de información mediante tecnologías avanzadas, como el uso de escáner láser (LiDAR), fotogrametría y drones, con el fin de obtener una representación precisa de la infraestructura existente. Este enfoque difiere parcialmente de los resultados obtenidos en la presente investigación, ya que el aplicativo desarrollado no se basa en tecnologías de captura masiva de datos, sino en el registro sistemático de información en campo mediante inspecciones directas. No obstante, ambos enfoques coinciden en un aspecto fundamental: la necesidad de contar con información confiable y actualizada como base para la gestión eficiente del mantenimiento.

De la integración de la información recopilada mediante el aplicativo con el modelo BIM para optimizar la gestión del mantenimiento en la Institución Educativa Ramón Castilla y Marquesado. Los resultados obtenidos evidencian que el modelado tridimensional desarrollado en Revit 2025 permitió consolidar una representación digital integral de la infraestructura educativa, incorporando los principales componentes estructurales, arquitectónicos, eléctricos y de mobiliario del Bloque I. Este modelo tridimensional constituye la base para la vinculación de la información recopilada en

campo mediante el aplicativo, permitiendo asociar parámetros técnicos y datos de mantenimiento a cada uno de los elementos que conforman la infraestructura.

En el caso de la disciplina estructural, el modelo BIM integró elementos fundamentales como muros, losas y cubiertas, lo que permitió establecer una base digital que facilita la incorporación de información técnica relacionada con el mantenimiento de la infraestructura. Esta representación tridimensional no solo permite visualizar la estructura del edificio, sino también vincular datos asociados al estado operativo, condición física y necesidades de intervención de cada componente, contribuyendo a una gestión más organizada y eficiente de los activos. De manera complementaria, el modelado arquitectónico permitió integrar los principales elementos constructivos del bloque educativo, incluyendo acabados como tarrajeo en muros y losas, contrapisos, pintura, cielorraso y elementos de carpintería metálica como puertas y ventanas. La estandarización de estos elementos dentro del modelo BIM facilita la incorporación de parámetros de mantenimiento asociados a cada componente arquitectónico, permitiendo identificar de manera más precisa las condiciones de los elementos evaluados y planificar intervenciones de mantenimiento de forma más eficiente. Asimismo, la integración de las instalaciones eléctricas dentro del modelo BIM permitió representar la distribución de canalizaciones, circuitos eléctricos, tableros de distribución, tomacorrientes y distintos tipos de luminarias presentes en el edificio. Esta representación digital facilita la identificación de los componentes eléctricos dentro de la infraestructura, permitiendo asociar la información recopilada mediante el aplicativo con los elementos específicos del sistema eléctrico, lo que contribuye a mejorar la planificación de las actividades de mantenimiento y la gestión de los equipos eléctricos instalados. Del mismo modo, el modelo BIM incorporó el mobiliario y equipamiento de las aulas, incluyendo sillas unipersonales, escritorio y silla del docente, pizarra acrílica, soporte para televisión y armarios de almacenamiento. La integración de estos elementos dentro del modelo permite visualizar la organización espacial de los ambientes educativos y gestionar de manera más eficiente los activos asociados al equipamiento institucional.

En este sentido, Sánchez (2021) enfatiza que BIM no debe ser entendido únicamente como una herramienta de modelado tridimensional, sino como una metodología orientada a la gestión de información, en la cual cada elemento del modelo contiene datos que aportan valor al proceso constructivo y de mantenimiento. Este enfoque

coincide directamente con los resultados obtenidos, ya que el modelo desarrollado en Revit 2025 no solo permitió representar geométricamente los componentes estructurales, arquitectónicos, eléctricos y de mobiliario, sino que además facilitó la incorporación de información técnica proveniente del aplicativo, convirtiendo al modelo en una base de datos digital para la gestión del mantenimiento. Asimismo, el autor ala que la adopción de BIM influye positivamente en la planificación y organización del mantenimiento, al integrar la información dentro de un entorno digital que permite mejorar la toma de decisiones. Este planteamiento es consistente con los resultados del presente estudio, donde la integración de datos en el modelo BIM permitió asociar parámetros como estado operativo, condición física y necesidades de intervención a cada elemento de la infraestructura, lo que contribuye a una gestión más eficiente de los activos. Sin embargo, a diferencia del enfoque de Sánchez, que se centra principalmente en la mejora de la gestión a nivel organizacional y de planificación, la presente investigación aporta un enfoque más aplicado al evidenciar la interoperabilidad entre un aplicativo de campo y el modelo BIM, lo que permite una actualización dinámica y continua de la información.

Consecuentemente en el presente estudio, la integración de la información recopilada mediante el aplicativo permitió generar paneles de control y análisis mediante la herramienta Power BI, lo que facilitó el monitoreo y evaluación del estado de los activos de la infraestructura educativa. A partir del análisis de los datos registrados, se identificó un total de 1376 activos dentro del sistema, de los cuales 1351 se encuentran operativos y 25 fuera de servicio, con un costo total estimado de intervención de S/ 52,187.96. Asimismo, el análisis de la condición física de los activos evidenció que 1310 activos se encuentran en buen estado, mientras que 37 presentan condición mala y 29 condición regular, lo que permite identificar de manera precisa el estado actual de la infraestructura evaluada. En cuanto al análisis por disciplinas, se identificó que el componente arquitectónico concentra la mayor cantidad de activos evaluados, con 724 elementos registrados, de los cuales 718 se encuentran operativos y 6 fuera de servicio, con un costo estimado de mantenimiento de S/ 37,947.96. En el caso de las instalaciones eléctricas, se registraron 65 activos, de los cuales 56 se encuentran operativos y 9 fuera de servicio, con un costo estimado de intervención de S/ 1,440.00. Por su parte, el mobiliario y equipamiento registró 217 activos, de los cuales 207 se encuentran operativos y 10 fuera de servicio, con un costo estimado de S/ 12,800.00. Estos resultados permiten identificar las áreas que requieren mayor atención dentro del proceso de mantenimiento de la

16 infraestructura educativa. Asimismo, el análisis de los activos en condición de fuera de servicio permitió identificar 25 elementos que requieren intervención, de los cuales 19 presentan criticidad alta y 6 criticidad media, evidenciando la necesidad de priorizar acciones de mantenimiento en aquellos componentes cuyo deterioro afecta de manera más significativa el funcionamiento de la infraestructura. El costo estimado de intervención para estos activos asciende a S/ 8,540.00, lo que proporciona una referencia económica para la planificación de las acciones de mantenimiento correctivo necesarias. Finalmente, el análisis de los activos operativos evidenció que 1351 elementos se encuentran funcionando adecuadamente, lo que representa la mayor parte de los activos evaluados dentro de la infraestructura. Asimismo, se identificó que la mayoría de estos activos presentan criticidad baja, lo que indica que su posible falla tendría un impacto reducido en el funcionamiento general del edificio. No obstante, se identificaron 12 activos con criticidad media y 6 con criticidad alta, los cuales requieren monitoreo constante para prevenir posibles fallas futuras y garantizar la continuidad del servicio educativo. En conjunto, los resultados obtenidos demuestran que la integración entre el aplicativo de recolección de datos, el modelo BIM y las herramientas de análisis de información permitió consolidar un sistema digital para la gestión del mantenimiento de la infraestructura educativa. Esta integración facilita la organización de la información, la identificación de activos críticos, la estimación de costos de intervención y la planificación de actividades de mantenimiento, contribuyendo a optimizar la gestión de los activos y mejorar la toma de decisiones en la administración de la infraestructura.

En este sentido, Venegas (2023) demuestra que la implementación de BIM mejora la calidad del servicio a nivel organizacional, evidenciando que el 96.3% de los encuestados percibe un nivel alto tanto en la aplicación de la metodología BIM como en la calidad del mantenimiento. Estos resultados coinciden con los hallazgos del presente estudio, en los cuales la integración de tecnologías digitales permite optimizar la gestión del mantenimiento. Sin embargo, mientras el estudio de Venegas se basa en percepciones mediante encuestas aplicadas a 54 profesionales, el presente estudio aporta un enfoque más robusto al trabajar con datos reales y medibles, tales como la identificación de 1376 activos, de los cuales 25 se encuentran fuera de servicio, así como la estimación de costos de intervención por un total de S/ 52,187.96. Asimismo, el autor plantea que la metodología BIM influye en dimensiones clave como la planificación, los tiempos de trabajo y los costos del mantenimiento, destacando que su correcta implementación permite optimizar

recursos y mejorar la eficiencia del servicio. Este planteamiento es consistente con los resultados obtenidos, donde el análisis mediante Power BI permitió identificar áreas críticas, como el componente arquitectónico con mayor carga de activos y costos, así como determinar la necesidad de intervención en activos con alta criticidad. No obstante, a diferencia del enfoque de Venegas, que evalúa estas dimensiones de forma general, la presente investigación profundiza en un nivel más detallado, permitiendo desagregar la información por disciplina, estado y criticidad, lo que facilita una planificación más precisa y estratégica del mantenimiento.

La hipótesis general plateada que si mediante el desarrollo de un aplicativo de recolección de información integrado al BIM se mejora la gestión del mantenimiento en un 70% en la Institución Educativa N°16001 Ramón Castilla y Marquesado. Los resultados obtenidos evidencian que la implementación del aplicativo permitió digitalizar el proceso de levantamiento de información en campo, facilitando el registro sistemático de los activos y sus condiciones operativas. A través del sistema desarrollado se logró registrar reportes de inspección, actualizar parámetros técnicos y mantener un historial de mantenimiento asociado a cada elemento de la infraestructura. En conjunto, se evidencia que la integración entre el aplicativo de recolección de datos, el modelo BIM y las herramientas de análisis digital contribuyó a mejorar la organización de la información, optimizar el monitoreo de los activos y fortalecer la toma de decisiones en la gestión del mantenimiento. En este sentido, los resultados obtenidos permiten aceptar la hipótesis general de la investigación, demostrando que la implementación de esta metodología tecnológica mejora significativamente la gestión del mantenimiento de la infraestructura educativa.

V. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- El desarrollo e implementación del aplicativo **App-Mantenimiento** permitió demostrar que la integración de tecnologías digitales como BIM, aplicaciones móviles y bases de datos en la nube constituye una alternativa eficiente para la gestión del mantenimiento de activos en infraestructuras educativas. La solución desarrollada permitió recolectar, registrar y gestionar información relacionada con los activos de la Institución Educativa Ramón Castilla y Marquesado, facilitando la identificación de equipos mediante códigos QR, el registro de incidencias de mantenimiento y la consulta de información técnica asociada a cada elemento del sistema. Asimismo, la interoperabilidad entre herramientas como Flutter, Firebase, Supabase, Dynamo, Revit y Power BI permitió consolidar un entorno tecnológico capaz de integrar el modelo BIM con una plataforma móvil de gestión, lo que contribuye a mejorar la organización de la información, la trazabilidad de las intervenciones de mantenimiento y el proceso de toma de decisiones.
- La implementación y evaluación del aplicativo móvil para la recolección de datos de infraestructura permitió optimizar el proceso de registro, organización y gestión de la información relacionada con el mantenimiento de los activos en la Institución Educativa Ramón Castilla y Marquesado. El sistema desarrollado facilitó la generación de reportes técnicos a partir de las inspecciones realizadas en campo, permitiendo registrar parámetros relevantes como el estado operativo, la condición física, el nivel de criticidad, el riesgo normativo, las acciones recomendadas, el costo estimado y la fecha de la última inspección de cada activo. Asimismo, la posibilidad de exportar la información recopilada hacia archivos en formato Excel permitió integrar los datos de mantenimiento con el modelo BIM desarrollado en Revit mediante el uso de Dynamo, lo que contribuyó a actualizar de manera automática los parámetros asociados a los elementos del modelo digital de la infraestructura. En este sentido, el aplicativo demostró ser una herramienta eficiente para sistematizar la información obtenida durante las inspecciones de campo, fortaleciendo el proceso de gestión del mantenimiento y mejorando la disponibilidad de información técnica para la toma de decisiones.
- La integración de la información recopilada mediante el aplicativo con el modelo BIM permitió optimizar la gestión del mantenimiento de la infraestructura de la Institución

Educativa Ramón Castilla y Marquesado. El desarrollo del modelo tridimensional en Revit 2025 permitió representar digitalmente los componentes estructurales, arquitectónicos, eléctricos y de mobiliario del Bloque I, constituyendo una base digital que facilita la incorporación y vinculación de información técnica obtenida durante las inspecciones de campo. Asimismo, la integración de los datos registrados en el aplicativo con herramientas de análisis como Power BI permitió generar paneles de control que facilitan el monitoreo del estado de los activos, identificando un total de 1376 elementos registrados en el sistema, de los cuales 1351 se encuentran operativos y 25 fuera de servicio. Este proceso permitió evaluar la condición física, el nivel de criticidad y los costos estimados de intervención, proporcionando información relevante para la planificación de actividades de mantenimiento. En consecuencia, la integración entre el aplicativo de recolección de datos, el modelo BIM y las herramientas de análisis digital demostró ser una metodología eficiente para mejorar la organización de la información, optimizar la gestión del mantenimiento y fortalecer la toma de decisiones en la administración de la infraestructura educativa.

- La implementación del aplicativo de recolección de información integrado con el modelo BIM permitió mejorar significativamente la gestión del mantenimiento de la infraestructura de la Institución Educativa N°16001 Ramón Castilla y Marquesado. En consecuencia, los resultados obtenidos permiten aceptar la hipótesis general de la investigación, evidenciando que la implementación del aplicativo integrado al BIM mejora la gestión del mantenimiento de la infraestructura educativa.

5.2. Recomendaciones

- Se recomienda implementar y ampliar el uso del aplicativo **App-Mantenimiento** en otras instituciones educativas o infraestructuras públicas, con el fin de fortalecer la gestión del mantenimiento mediante herramientas digitales interoperables basadas en BIM.

Asimismo, se sugiere continuar desarrollando nuevas funcionalidades dentro del sistema, tales como la integración con sistemas de mantenimiento predictivo, el uso de sensores IoT para monitoreo en tiempo real de los activos y la ampliación de los módulos de análisis mediante plataformas de inteligencia de negocios. Estas mejoras permitirían optimizar aún más la gestión de la infraestructura, facilitar la planificación de intervenciones de mantenimiento y contribuir a una administración más eficiente y sostenible de los activos físicos.

- Se sugiere continuar mejorando el aplicativo mediante la incorporación de funcionalidades que permitan automatizar aún más el procesamiento de la información, como la sincronización directa entre el aplicativo y el modelo BIM sin necesidad de exportación intermedia, así como la integración con plataformas de análisis de datos que faciliten la evaluación del estado de los activos y la planificación de intervenciones de mantenimiento de manera más eficiente.
- Se recomienda implementar un programa de mantenimiento preventivo basado en la información generada por el aplicativo y el modelo BIM, con el fin de anticipar posibles fallas en los activos de la infraestructura educativa y garantizar su adecuado funcionamiento a lo largo del tiempo. A partir de los datos registrados sobre estado operativo, condición física y nivel de criticidad de los elementos evaluados, es posible establecer cronogramas periódicos de inspección y mantenimiento que permitan intervenir oportunamente antes de que se presenten fallas mayores. La adopción de estrategias de mantenimiento preventivo contribuirá a prolongar la vida útil de los activos, optimizar el uso de los recursos económicos destinados a mantenimiento y asegurar condiciones adecuadas de funcionamiento y seguridad en la infraestructura de la Institución Educativa.
- Se sugiere ampliar el alcance de estudios similares en diferentes tipos de infraestructuras y contextos institucionales, con el fin de comparar resultados y validar la efectividad de los aplicativos de gestión de mantenimiento basados en BIM como herramientas para mejorar la eficiencia en la administración de activo

REFERENCIAS BIBLIOGRÁFICAS

Abanto-Oblitas, R. M. (2023). *Plan de operación y mantenimiento preventivo para infraestructura escolar pública a través de un sistema de gestión BIM basada en la ISO 19650-3* [Tesis de pregrado, Pontificia Universidad Católica del Perú]. Archivo digital. <http://hdl.handle.net/20.500.12404/24361>

Amórtegui-Guzmán, E. (2021). *Análisis comparativo de la implementación de la metodología BIM en un proyecto de vivienda en la ciudad de Bogotá* [Tesis de

pregrado, Universidad Francisco José de Caldas]. Archivo digital. <http://hdl.handle.net/11349/30211>

Arias, F. (2016). El proyecto de investigación: Introducción a la metodología científica (6a ed.). Editorial Episteme.

Barrera-Pablos, S. A., & Guerrero-Pérez, J. M. (2023). *Propuesta de dotación y gastos de mantenimiento mediante la metodología bim del edificio aulas oriente de la universidad Francisco de Paula Santander-sede Cúcuta* [Tesis de pregrado, Universidad Francisco de Paula Santander]. Archivo digital. <https://repositorio.ufps.edu.co/handle/ufps/6841>

Bellota-Gonzales, L. W. (2022). *Análisis de la influencia en la detección de interferencias y compatibilidad de metrados entre planos estructurales e instalaciones sanitarias, utilizando la metodología tradicional CAD2D y la metodología BIM en el diseño de dos centros de salud* [Tesis de pregrado, Universidad Andina del Cusco]. Archivo digital. <https://hdl.handle.net/20.500.12557/5260>

Coupry , C., Noblecourt , S., Richard, P., Baudry, D., & Bigaud, D. (2021). BIM-Based Digital Twin and XR Devices to Improve Maintenance Procedures in Smart Buildings: A Literature Review. *Applied Siences*, 11(15). Obtenido de <https://doi.org/10.3390/app11156810>

Dávila-Figero, V. H., Ramírez-Peña, L. T., & Suarez-Prieto, J. E. (2022). *Desarrollo de una guía metodológica para la gerencia de proyectos de construcción de edificaciones sostenibles en Colombia integrando BIM – etapas de planeación y monitoreo y control* [Tesis de postgrado, Escuela Colombiana de Ingeniería Julio Garavito]. Archivo digital. <https://repositorio.escuelaing.edu.co/handle/001/2118>

Gómez-Ruiz, J. S., Perilla-Niño, A. F., & Vanegas-Torres, J. A. (2021). *Servicio de mantenimiento integral en infraestructura clínica (I.P.S) mediante la metodología facility Management y B.I.M* [Tesis de pregrado, Universidad Colegio Mayor de Cundinamarca]. Archivo digital. <https://repositorio.unicolmayor.edu.co/handle/unicolmayor/5384>

Gonzales-Escobar, I. C., & Rodríguez-Vásquez, Y. (2021). *Metodología BIM para mejorar la planificación y eficiencia de procesos en el área estructural de una constructora,*

- Cajamarca -2020* [Tesis de pregrado, Universidad César Vallejo]. Archivo digital. <https://hdl.handle.net/20.500.12692/64067>
- Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). Metodología de la investigación (6a ed.). McGraw-Hill Education.
- Hernández-Dávalos, S. R., & Merino-Adriazola, A. L. (2022). *Propuesta para la implementación de BIM 7D en el mantenimiento de una edificación en la UNALM* [Tesis de pregrado, Universidad Tecnológica del Perú]. Archivo digital. <https://hdl.handle.net/20.500.12867/7397>
- Matarrita Chaves, R. M., & Gera María, G. G. (2020). Building Information Modeling (BIM) en la gestión de mantenimiento hospitalario. *Universidad Autónoma Metropolitana*. Obtenido de <http://hdl.handle.net/11191/8009>
- Mendez-Asencio, R. M. (2020). *Complementación de modelos BIM en programa mantenimiento de infraestructura hospitalaria Villa El Salvador 2018* [Tesis de posgrado, Universidad San Pedro]. Archivo digital. <http://repositorio.usanpedro.edu.pe/handle/USANPEDRO/14701>
- Palomino Ojeda, J. M., Quiñones Huatangari, L., Cayatopa Calderon, B. A., Piedra Tineo, J. L., Apaza Panca, C. Z., & Milla Pino, M. E. (2024). Estimation of the Physical Progress of Work Using UAV and BIM. *10(2)*. Obtenido de 10.28991/CEJ-2024-010-02-02
- Sánchez-Manayay, M. (2022). *Metodología BIM en la mejora del mantenimiento preventivo y correctivo de edificios en la empresa ASPERSUD, Lima 2021* [Tesis de posgrado, Universidad César Vallejo]. Archivo digital. <https://hdl.handle.net/20.500.12692/68362>
- Sánchez - Manayay, M. (2022). *Implementación de una gestión de calidad utilizando la metodología BIM para mantenimiento de edificio en el distrito de Pueblo Libre, año 2021* [Trabajo de grado, Universidad Privada del Norte]. Archivo digital. <https://repositorio.upn.edu.pe/item/16dd6829-d39e-4996-9fe3-fa10f810d0e73>
- Suarez-Rivas, V. F., & Salazar-Chumioque, J. D. (2023). *Procedimiento para la mejora de la etapa de operación y mantenimiento de colegios usando BIM en Lima* [Trabajo de suficiencia profesional, Universidad Peruana de Ciencias Aplicadas]. Archivo digital. <http://hdl.handle.net/10757/670123>

- Venegas-Vergara, M. A. (2023). *Metodología BIM y la calidad de servicio de mantenimiento de infraestructura en una entidad pública, Lima, 2023* [Tesis de pregrado, Universidad Cesar Vallejo]. Archivo digital. <https://repositorio.ucv.edu.pe/handle/20.500.12692/138490>
- Xianfei, Y., Liu, H., Chen, Y., Wang, Y., & Al-Hussein, M. (2020). A BIM-based framework for operation and maintenance of utility tunnels. *Tunnelling and Underground Space Technology*, 97. doi:<https://doi.org/10.1016/j.tust.2019.103252>
- Yopla-Ocas, J. C., & Zavaleta-Hoyos, D. (2021). *Incompatibilidades e interferencias determinadas con la metodología BIM en el proyecto mercado de abastos – Los Baños del Inca - Cajamarca* [Tesis de pregrado, Universidad Privada del Norte]. Archivo digital. <https://hdl.handle.net/11537/28626>
- Zamora-Pozo, R. (2023). *Optimización del proceso de preparación y documentación de un proyecto de edificación para la etapa de operación y mantenimiento, mediante la metodología BIM* [Tesis de pregrado, Universidad Privada del Norte]. Archivo digital. <https://hdl.handle.net/11537/34244>
- Zhao, J., Feng, H., Chen, Q., & Garcia de Soto, B. (2022). Developing a conceptual framework for the application of digital twin technologies to revamp building operation and maintenance processes. *Journal of Building Engineering*, 49. Obtenido de <https://doi.org/10.1016/j.job.2022.104028>