Universidad Autónoma De Yucatán

Facultad de Matemáticas

Licenciatura en Ingeniería de Software

Construcción de Software

Profesor: M.T.I. Edwin Jesús León Bojórquez

Entrega de documento final del proyecto: Agenda Escolar con Sistema de Calificación Ponderada

Equipo:

Campos García Karen Elizabeth

Castro Santeliz Héctor Alejandro

García Avilés Joseph Antonio

Pérez Zumárraga Rubén Alejandro

Fecha de entrega: 3 de diciembre de 2025

Contenido

Modelado del diseño de la aplicación (Diagrama de clases) 5

Checklist de las prácticas de construcción 5

1. Variables y tipos de datos 5

2. Nombres de variables y constantes 6

3. Organización y formato del código 7

4. Estructuras de control 8

5. Construcción de procedimientos 9

6. Construcción de clases y módulos 11

7. Singleton 12

8. Comentarios y documentación 12

9. Arquitectura del proyecto 13

Pruebas de Aceptación (Especificación de Requerimientos vs Funcionalidad del Software) 14

1. Requisitos Funcionales 14

1.1 Gestión de actividades 14

1.2 Gestión de materias 15

1.3 Gestión de tipos de actividades 17

1.4 Estados de actividades 18

1.5 Vistas del sistema 18

3.1.6 Filtrado y búsqueda 20

3.1.7 Sistema de calificación - configuración 21

1.8 Sistema de calificación - cálculos automáticos 23

1.9 Sistema de calificación - análisis y proyecciones 25

1.10 Visualización de Calificaciones/Materia 27

2. Requisitos no funcionales 28

2.1 Persistencia y almacenamiento de datos 28

2.2 Usabilidad e interfaz de usuario 29

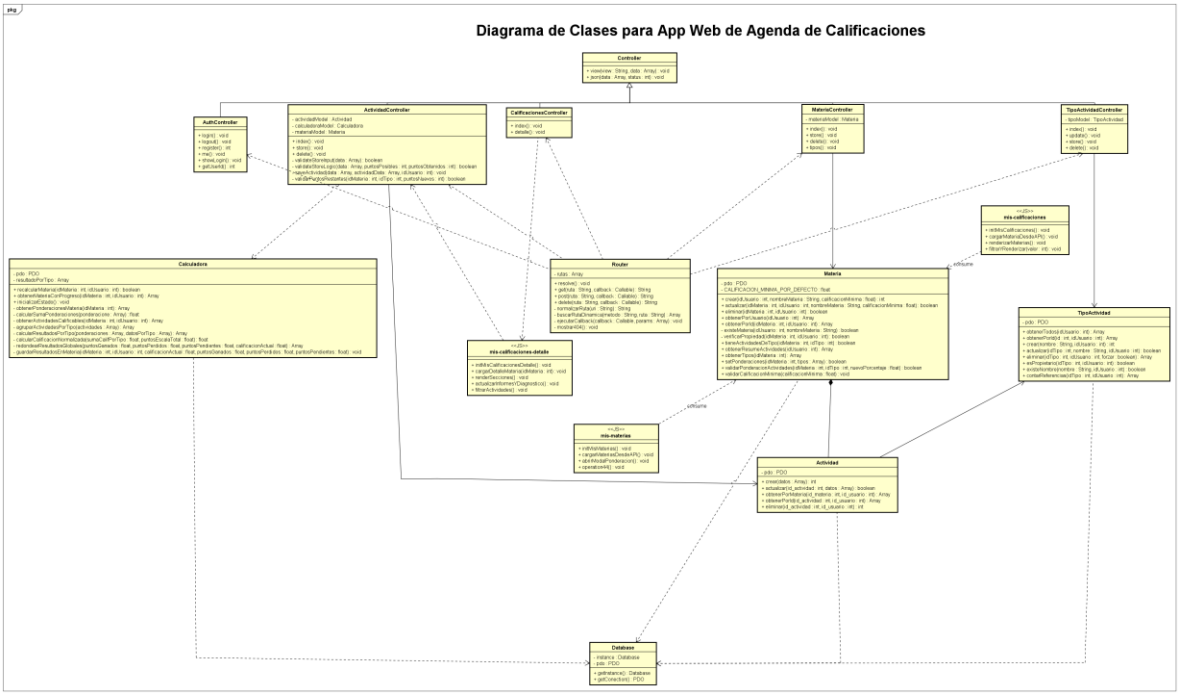
2.3 Rendimiento 30

2.4 Confiabilidad y Disponibilidad 31

Evidencias de las métricas de calidad del código. 32
1. Seguridad 32
1.1 Overview 32
1.2 Rating de seguridad 32
1.3 Número de vulnerabilidades 33
2. Confiabilidad 33
2.1 Overview 33
2.2 Número de Bugs 34
2.3 Rating de confiabilidad 34
2.4 Esfuerzo de remediación de confiabilidad 35
3. Mantenibilidad 35
3.1 Overview 35
3.2 Número de code smells 36
3.3 Deuda técnica 36
3.4 Radio de deuda técnica 36
3.5 Rating de mantenibilidad 37
4. Análisis de seguridad (riesgos potenciales) 38
4.1 Número de hotspots 38
4.2 Rating 38
5. Duplicación 39
5.1 Overview 39
5.2 Líneas duplicadas 39
5.3 Bloques duplicados 39
5.4 Archivos duplicados 40
6. Tamaño 41
6.1 Líneas de código 41
6.2 Líneas totales 41
6.3 Número de sentencias ejecutables 42
6.4 Número de funciones 42

6.5 Número de clases 43
6.6 Número de archivos 44
6.7 Líneas de comentarios 45
6.8 Porcentaje de líneas comentadas 46
7. Complejidad ciclomática 47
8. Complejidad cognitiva 48
9. Overall.....48
Evidencias de pruebas unitarias 50
Javascript.....50
PHP (Backend) 51

Modelado del diseño de la aplicación (Diagrama de clases)



Checklist de las prácticas de construcción

1. Variables y tipos de datos

Validado	Característica
X	Las variables están declaradas explícitamente
X	Las variables están inicializadas antes de usarse
X	Las variables se inicializan cerca de donde se usan
X	Se revisa cuando una variable debe reinicializarse
X	El alcance (scope) de cada variable es el mínimo necesario

X	El tiempo de vida de cada variable es lo más corto posible
X	Las referencias a cada variable están cerca entre sí (span corto)
X	No se guarda persistencia accidental en variables temporales
X	Se validan los parámetros de entrada
X	Se evitan números mágicos (valores literales duros)
X	Se evitan cadenas mágicas (strings literales con significado especial sin constante)
X	Se usan constantes con nombre para valores especiales
X	Las conversiones de tipo son explícitas
X	No se comparan tipos mezclados sin intención
X	Se atienden advertencias del IDE
X	No se comparan flotantes por igualdad exacta cuando importa la precisión
X	Las cadenas se tratan con precaución (índices, longitud, substrings)
X	No hay errores off-by-one
X	Se respetan los límites de los arreglos

2. Nombres de variables y constantes

Validado	**Característica**

X	Nombres claros y descriptivos
X	Convención de nombres consistente
X	Sufijos como Total, Count, Max, Min se usan correctamente
X	Uso coherente de pares de opuestos (start/end, open/close)
X	Índices i, j, k solo se usan en bucles simples
X	Nombres significativos para bucles anidados
X	Banderas booleanas con nombres positivos
X	No se usan variables temporales vagas (temp, aux, data2)
X	Constantes con nombres significativos
X	No hay abreviaturas confusas o inconsistentes
X	No hay nombres demasiado parecidos que generen confusión
X	No se mezclan idiomas en los nombres

3. Organización y formato del código

Validado	Característica
X	El código fluye de forma lógica de arriba hacia abajo
X	Las sentencias relacionadas están agrupadas
X	La sangría es consistente en todo el proyecto

X	Existe separación vertical entre bloques conceptuales
X	Se usan espacios para mejorar legibilidad
X	Las funciones relacionadas están agrupadas
X	El formato es homogéneo en todos los archivos

4. Estructuras de control

Validado	Característica
X	El caso normal va en el bloque if
X	Las cláusulas else están justificadas
X	Se simplifican condiciones complejas mediante funciones booleanas
X	Se usa switch cuando mejora la claridad
X	Los case están ordenados lógicamente
X	No hay fall-through accidental entre case
X	El tipo de bucle elegido es adecuado para el caso
X	Se usa for cuando se conoce la cantidad de iteraciones o depende de un índice
X	Se usa while cuando la repetición depende de una condición previa
X	Se usa do…while cuando se necesita ejecutar al menos una vez

X	Se usa foreach / for…of / for…in para colecciones según corresponda
X	La inicialización del bucle está en la cabecera o justo antes
X	El bucle solo tiene una entrada
X	No hay bucles vacíos
X	Las condiciones de terminación son claras
X	No se manipulan índices de forma confusa dentro del bucle
X	El anidamiento de bucles es limitado
X	Los bucles son lo suficientemente cortos para leerse completos
X	Los índices tienen nombres significativos en bucles anidados
X	Los índices están limitados al ámbito del bucle
X	break y continue se usan con prudencia

5. Construcción de procedimientos

Validado	Característica
X	Cada función tiene un propósito principal claro (alta cohesión)
X	Las funciones no mezclan niveles de abstracción distintos
X	Las funciones son cortas y legibles
X	No hay lógica duplicada dentro del proyecto

X	Las funciones ocultan detalles internos
X	Las operaciones complejas se encapsulan en funciones auxiliares
X	Las funciones favorecen reutilización y mantenimiento
X	Las expresiones booleanas complejas se extraen a funciones
X	Las funciones demasiado largas se dividen adecuadamente
X	Los nombres de funciones describen exactamente lo que hacen
X	Los procedimientos usan verbo + objeto (guardarUsuario)
X	Las funciones indican en su nombre el valor que devuelven
X	Los parámetros siguen un orden consistente (entrada → modificación → salida)
X	El número de parámetros es razonable (idealmente máximo 5–7)
X	No hay parámetros sin uso
X	Los parámetros tienen nombres significativos
X	Se documentan precondiciones importantes
X	Se documentan postcondiciones cuando aplica
X	Todas las rutas de ejecución retornan coherentemente

X	No se devuelven referencias peligrosas a estructuras internas

6. Construcción de clases y módulos

Validado	Característica
X	Cada clase o módulo tiene una responsabilidad clara
X	La implementación interna está oculta (encapsulación)
X	La interfaz pública es coherente con el propósito de la clase
X	No se exponen propiedades internas sin necesidad
X	No existen clases con demasiadas responsabilidades
X	El acoplamiento entre clases/módulos es bajo
X	No se encadenan múltiples accesos profundos innecesariamente
X	La herencia se usa solo cuando un objeto realmente es un subtipo del otro
X	Se prefiere composición sobre herencia
X	Los constructores inicializan todo lo necesario
X	Los métodos tienen alta cohesión con los datos de la clase

X	El número de métodos públicos es razonable
X	Las dependencias se inyectan cuando es posible
X	No se sobrescriben métodos solo para dejarlos vacíos

7. Singleton

Validado	Característica
X	El constructor es privado o el acceso está controlado
X	Existe un único punto para obtener la instancia
X	La instancia única está garantizada

8. Comentarios y documentación

Validado	Característica
X	Los comentarios explican el por qué, no el qué
X	No se repite en comentarios lo que el código ya expresa
X	Los comentarios están cerca del código que describen
X	Las funciones importantes tienen una descripción breve
X	Los parámetros importantes están documentados
X	Los bloques complejos tienen un comentario previo

X	No hay comentarios desactualizados o engañosos
X	Se documentan decisiones no obvias
X	Los comentarios no se usan como excusa para mal código
X	Se documentan variables globales cuando existen
X	Los archivos pueden tener encabezados con propósito y autor
X	Se usan etiquetas TODO, FIXME y NOTE de forma ordenada

9. Arquitectura del proyecto

Validado	Característica
X	Hay separación clara entre capas (HTML/CSS, JS, PHP, datos)
X	La estructura de carpetas es coherente
X	No hay archivos demasiado grandes con demasiada responsabilidad
X	Los nombres de archivos reflejan su función
X	No hay dependencias circulares entre módulos
X	Existen servicios/helpers/utilidades reutilizables

Pruebas de Aceptación (Especificación de Requerimientos vs Funcionalidad del Software)

1. Requisitos Funcionales

1.1 Gestión de actividades

ID	Descripción	Validado	Observaciones
RF-001	El sistema permitirá crear actividades especificando obligatoriamente: nombre, materia, tipo, fecha de entrega/realización y estado. Opcionalmente: campos de calificación.	X	
RF-002	El sistema permitirá editar cualquier campo obligatorio de una actividad existente en cualquier momento.	X	
RF-003	El sistema permitirá eliminar actividades.	X	
RF-004	El sistema mostrará un mensaje de advertencia al intentar eliminar una actividad.	X	Se determinó que mostrar advertencias en cada eliminación resultaba

			intrusivo para el usuario, por lo que se implementó un control de confirmación global que aplica a todas las actividades.
RF-005	El sistema validará que todos los campos obligatorios estén llenos antes de guardar una actividad nueva o editada.	X	

1.2 Gestión de materias

ID	Descripción	Validado	Observaciones
RF-006	El usuario podrá crear todas las materias y tipos de actividad asociados que necesite, sin ninguna restricción en la cantidad.	X	
RF-007	El usuario podrá configurar la calificación aprobatoria de las materias y sus tipos.	X	

RF-008	El sistema permitirá editar el nombre de una materia.	X	
RF-009	El sistema permitirá configurar la calificación mínima aprobatoria por materia (por defecto 70). Valores comunes: 60, 70, 80.	X	
RF-010	Al intentar eliminar una materia, el sistema verificará si tiene actividades asociadas. Si las tiene, mostrará una advertencia y solicitará confirmación explícita.	X	
RF-011	Al eliminar una materia con actividades asociadas, el sistema debe eliminar las actividades según confirmación del usuario.	X	
RF-012	El sistema permitirá eliminar tipos de actividad personalizados asociados a una materia, siempre que el usuario otorgue una confirmación explícita. Antes de proceder, el sistema mostrará una advertencia clara sobre las consecuencias de la eliminación.	X	

1.3 Gestión de tipos de actividades

ID	Descripción	Validado	Observaciones
RF-013	El sistema incluirá tipos predeterminados que no pueden eliminarse, por ejemplo: Tarea, Examen, Proyecto.	X	
RF-014	El usuario podrá registrar tantos tipos como necesite, sin restricción de cantidad.	X	
RF-015	El sistema permitirá eliminar los tipos personalizados, pero no predeterminados, siempre y cuando haya confirmación explícita del usuario.	X	
RF-016	El sistema permitirá eliminar tipos personalizados a nivel global, siempre que el usuario proporcione una confirmación explícita antes de proceder.	X	
RF-017	Al intentar eliminar un tipo que tenga actividades o ponderaciones asociadas, el sistema mostrará una advertencia	X	

	indicando que dichos elementos también serán eliminados.		

1.4 Estados de actividades

ID	Descripción	Validado	Observaciones
RF-018	Cada actividad tendrá uno de tres estados posibles: "Pendiente" (por defecto al crear), "En proceso" o "Listo".	X	
RF-019	El sistema permitirá cambiar el estado de una actividad en cualquier momento mediante controles visibles (dropdown, botones o similar).	X	
RF-020	Cada estado tendrá un indicador visual distintivo (color y/o icono) consistente en todas las vistas del sistema.	X	

1.5 Vistas del sistema

ID	Descripción	Validado	Observaciones
RF-021	El sistema proporcionará una vista de tabla que muestre todas las actividades	X	

	en formato tabular con columnas: nombre, materia, tipo, fecha, estado.		
RF-022	El sistema proporcionará una vista de materias que muestre los tipos y ponderaciones correspondientes asociados a cada materia.	X	
RF-023	El sistema proporcionará una vista de calificaciones que muestre el progreso académico detallado por cada materia con desglose por tipo.	X	
RF-024	El sistema permitirá navegar entre las vistas mediante botones, pestañas o menú claramente identificados.	X	
RF-025	El sistema proporcionará una vista de calendario que muestre las actividades distribuidas por fecha de entrega/realización en formato mensual.		
RF-026	En la vista de calendario, si varias actividades coinciden en la misma fecha, todas deben ser visibles o		

	accesibles mediante algún mecanismo (lista, scroll, etc.).		

3.1.6 Filtrado y búsqueda

ID	Descripción	Validado	Observaciones
RF-027	La vista de tabla permitirá filtrar actividades por materia, estado o tipo.	X	
RF-028	El sistema permitirá aplicar filtros combinados (materia + tipo + estado) simultáneamente.	X	
RF-029	El sistema permitirá visualizar las actividades calificables.	X	
RF-030	El sistema permitirá buscar actividades por nombre mediante un campo de búsqueda de texto.	X	
RF-031	El sistema proporcionará un botón o enlace para limpiar todos los filtros activos y restaurar la vista completa.	X	

3.1.7 Sistema de calificación - configuración

ID	Descripción	Validado	Observaciones
RF-032	El sistema considerará una actividad como “calificable” únicamente cuando el campo “Puntos posibles” tenga un valor numérico almacenado (es decir, no sea nulo).	X	
RF-033	El sistema excluirá automáticamente de todos los cálculos de progreso, ponderaciones y diagnósticos a las actividades cuyo campo “Puntos posibles” sea nulo, tratándolas como no calificables.	X	
RF-034	El sistema interpretará un valor de “Puntos obtenidos = 0” en una actividad calificable como un resultado válido que representa calificación capturada, diferenciándolo de un valor nulo, que indica calificación pendiente.	X	
RF-035	El sistema considerará un valor nulo en el campo “Puntos obtenidos” de una	X	

	actividad “calificable” como parte de los puntos pendientes.		
RF-036	El sistema validará que los "Puntos obtenidos" no excedan los "Puntos posibles", mostrando mensaje de error si se viola esta regla.	X	
RF-037	El sistema validará que los campos de puntos contengan solo números válidos (enteros o decimales positivos).	X	
RF-038	Por cada materia, el sistema permitirá configurar el porcentaje de ponderación para cada tipo de actividad disponible.	X	
RF-039	El sistema validará que la suma de todos los porcentajes de ponderación por materia sea exactamente 100% antes de permitir guardar la configuración.	X	
RF-040	Si los porcentajes no suman 100%, el sistema mostrará un mensaje de error.	X	

RF-041	El sistema permitirá guardar configuraciones de ponderación con tipos en 0% (por ejemplo, si una materia no tiene exámenes).	X	

1.8 Sistema de calificación - cálculos automáticos

ID	Descripción	Validado	Observaciones
RF-042	El sistema considerará una actividad como “calificada” cuando tenga puntos posibles > 0 y el campo “Puntos obtenidos” tenga un valor numérico almacenado (incluyendo 0, pero no nulo).	X	
RF-043	Para cada tipo de actividad en una materia, el sistema calculará su contribución parcial a la calificación como el promedio de (suma de “Puntos obtenidos” / suma de “Puntos posibles” de las actividades calificadas de ese tipo), escalado por la ponderación asignada al tipo.	X	
RF-044	El sistema calculará la calificación actual de cada materia sumando las	X	

	contribuciones de todos los tipos de actividad y normalizándolas en una escala de 0 a 100, de acuerdo con la suma de sus ponderaciones.		
RF-045	El sistema calculará y mostrará los "Puntos ganados": suma total de puntos obtenidos en todas las actividades calificadas de la materia.	X	
RF-046	El sistema calculará y mostrará los "Puntos perdidos": diferencia entre puntos posibles y obtenidos en actividades calificadas.	X	
RF-047	El sistema calculará los “Puntos futuros” como la diferencia entre la ponderación asignada al tipo de actividad y la suma de sus “Puntos posibles” definidos, cuando dicha suma sea menor que la ponderación. Esta diferencia representará puntos que aún pueden obtenerse mediante actividades no creadas.	X	
RF-048	El sistema calculará y almacenará los “Puntos pendientes” como la parte de la	X	

	escala total aún en juego, formada por: (a) los puntos de actividades definidas, pero sin “Puntos obtenidos” y (b) los “Puntos futuros”.		
RF-049	El sistema recalculará automáticamente todas las calificaciones y métricas cada vez que se modifiquen: puntos de actividades, ponderaciones.	X	
RF-050	Si un tipo de actividad tiene ponderación asignada pero no cuenta con actividades calificadas, su contribución completa se considerará dentro de los “Puntos pendientes”, distribuidos entre puntos pendientes definidos y puntos futuros según existan o no actividades creadas.	X	

1.9 Sistema de calificación - análisis y proyecciones

ID	Descripción	Validado	Observaciones
RF-051	El sistema calculará la proyección mínima: calificación actual considerando que se pierden todos los puntos pendientes.	X	

RF-052	El sistema calculará la proyección máxima: calificación actual considerando que se obtienen todos los puntos pendientes.	X	
RF-053	El sistema determinará el estado académico de cada materia basándose en la calificación mínima aprobatoria configurada: "Aprobado" (≥mínimo), "En riesgo" (mínimo-10 hasta mínimo-1), "Reprobado" (<mínimo-10).	X	
RF-054	El sistema calculará cuántos puntos de los pendientes necesita el usuario para alcanzar exactamente la calificación mínima aprobatoria.	X	
RF-055	Si la calificación actual ya supera la mínima aprobatoria, el sistema mostrará un mensaje indicando que ya está aprobado.	X	
RF-056	Si es matemáticamente imposible aprobar (incluso obteniendo todos los puntos pendientes), el sistema mostrará un mensaje indicando esta situación.	X	

1.10 Visualización de Calificaciones/Materia

ID	Descripción	Validado	Observaciones
RF-057	La vista de calificaciones mostrará para cada materia: nombre, calificación actual (%), estado académico, desglose por tipo con porcentajes.	X	
RF-058	La vista de calificaciones mostrará el resumen: puntos ganados, perdidos, pendientes, proyección mínima-máxima, y puntos necesarios para aprobar.	X	
RF-059	La vista de calificaciones mostrará el desglose detallado por tipo incluyendo todas las actividades calificables con sus puntos individuales.	X	
RF-060	La vista de calificaciones siempre mostrará todas las actividades "calificables"	X	
RF-061	El usuario podrá distinguir de alguna forma las calificaciones “calificables”	X	

RF-062	La vista de materia incluirá un botón o enlace para acceder a la configuración de ponderación de cada materia.	X	

2. Requisitos no funcionales

2.1 Persistencia y almacenamiento de datos

ID	Descripción	Validado	Observaciones
RNF-001	El sistema almacenará permanentemente todos los datos (actividades, materias, tipos, configuraciones, puntos) en una base de datos relacional.	X	
RNF-002	El sistema garantizará la integridad referencial entre actividades, materias y tipos mediante llaves foráneas y restricciones de base de datos.	X	
RNF-003	Las operaciones que afecten múltiples registros relacionados deben ejecutarse como transacciones atómicas (todo o nada).	X	

RNF-004	El sistema guardará automáticamente los cambios sin requerir acción manual del usuario (auto-guardado tras confirmación).	X	
RNF-005	Los datos deben persistir entre sesiones (no se pierden al cerrar la aplicación).	X	

2.2 Usabilidad e interfaz de usuario

ID	Descripción	Validado	Observaciones
RNF-006	Los controles e indicadores visuales deben ser consistentes en todas las vistas del sistema.	X	
RNF-007	El sistema debe proporcionar retroalimentación visual inmediata para todas las acciones del usuario (guardado, error, confirmación).	X	
RNF-008	Los mensajes de error deben ser claros, específicos y proporcionar sugerencias de solución cuando sea posible.	X	

RNF-009	Las acciones críticas deben requerir confirmación explícita del usuario.	X	
RNF-010	El sistema debe usar terminología clara y familiar para estudiantes (evitar jerga técnica innecesaria).	X	
RNF-011	Los campos de formulario deben incluir etiquetas claras y, cuando sea relevante, ejemplos o tooltips explicativos.	X	

2.3 Rendimiento

ID	Descripción	Validado	Observaciones
RNF-012	Los cálculos de calificaciones deben completarse y actualizarse en menos de 1 segundo tras modificar datos.	X	
RNF-013	El cambio entre vistas (tabla, calendario, calificaciones) debe ser inmediato (menos de 0.5 segundos).	X	

RNF-014	El sistema debe manejar eficientemente hasta 100 actividades totales sin degradación perceptible del rendimiento.	X	

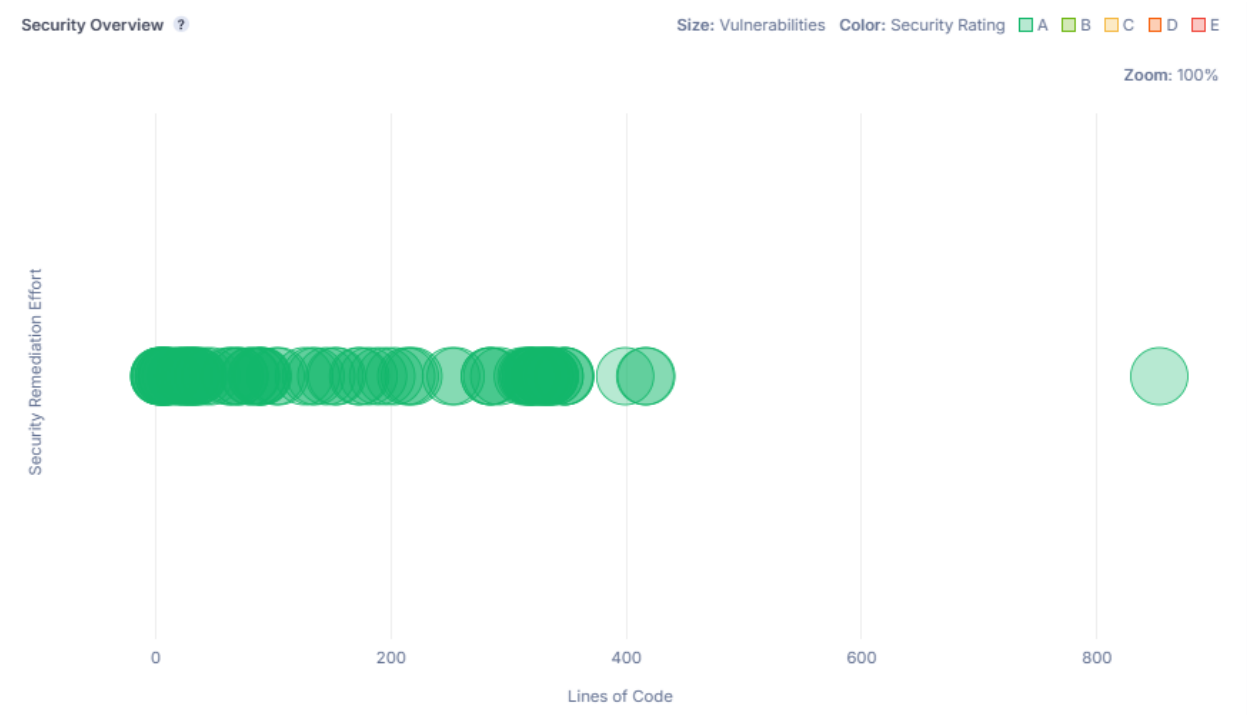
2.4 Confiabilidad y Disponibilidad

ID	Descripción	Validado	Observaciones
RNF-015	El sistema debe validar los datos antes de almacenarlos para garantizar consistencia de la base de datos.	X	
RNF-016	El sistema debe manejar errores de base de datos sin perder datos del usuario y mostrando mensajes informativos.		Cumple parcialmente en el backend (controladores que capturan excepciones y devuelven JSON de error)

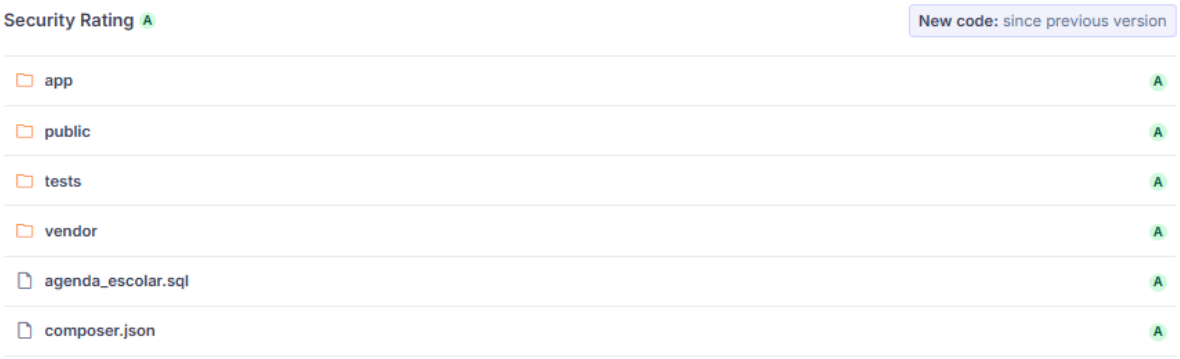
Evidencias de las métricas de calidad del código.

1. Seguridad

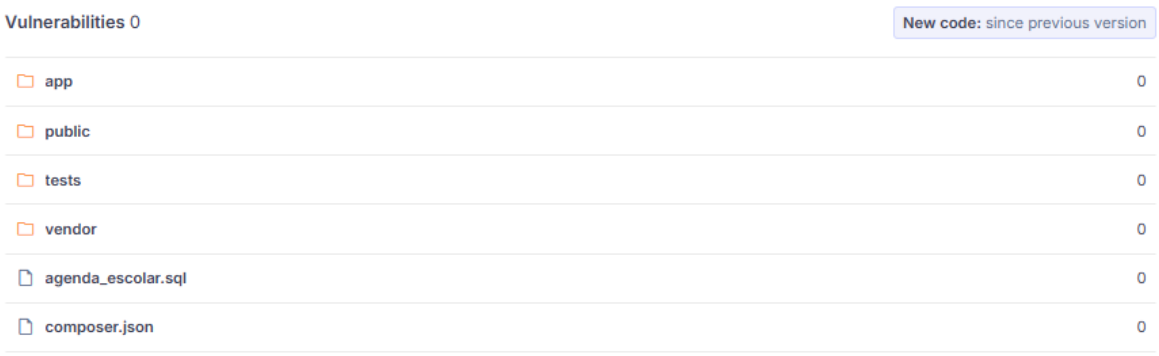
1.1 Overview



1.2 Rating de seguridad

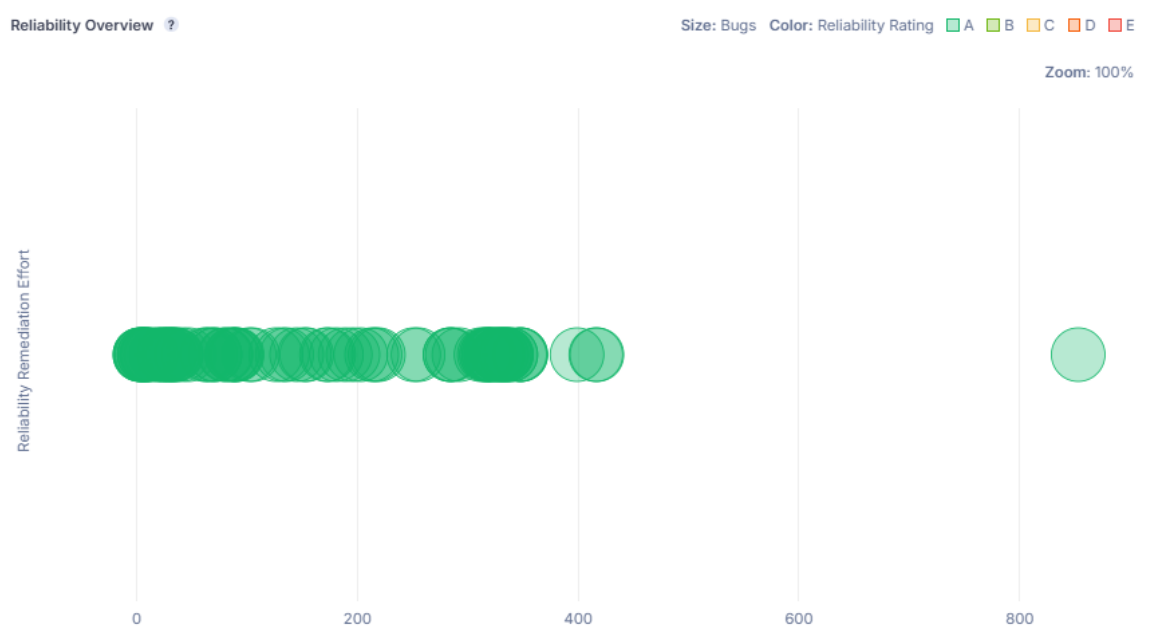


1.3 Número de vulnerabilidades

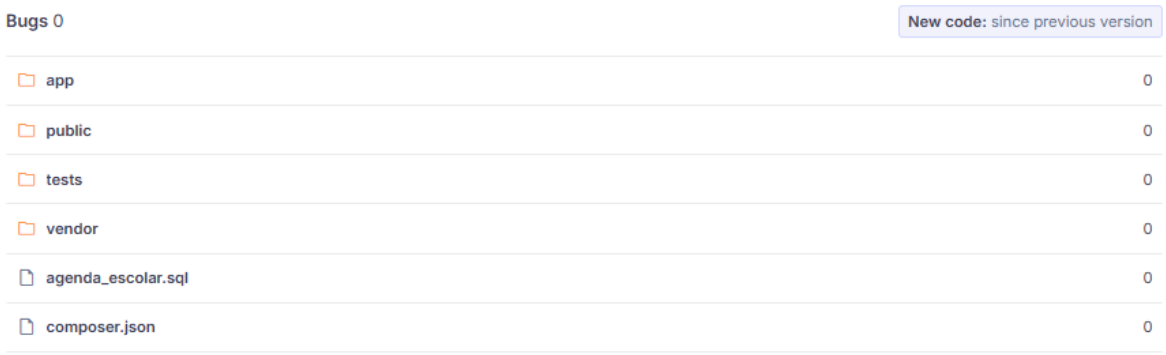


2. Confiabilidad

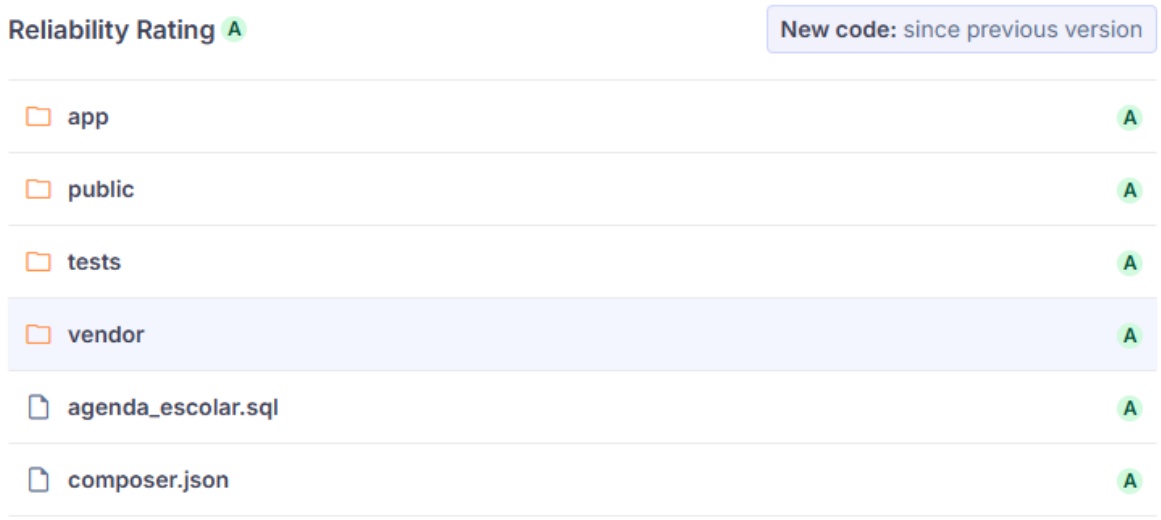
2.1 Overview



2.2 Número de Bugs



2.3 Rating de confiabilidad



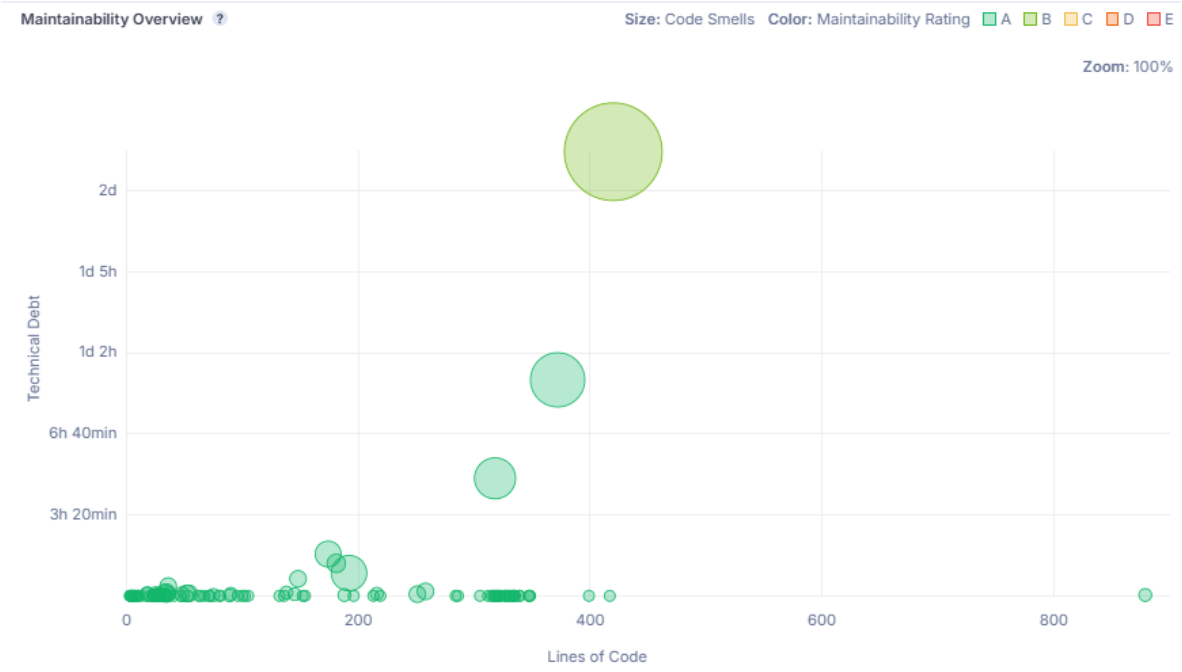
2.4 Esfuerzo de remediación de confiabilidad

Reliability Remediation Effort 0 | **New code:** since previous version

app	0
public	0
tests	0
vendor	0
agenda_escolar.sql	0
composer.json	0

3. Mantenibilidad

3.1 Overview



3.2 Número de code smells

Code Smells 158 — **New code:** since previous version

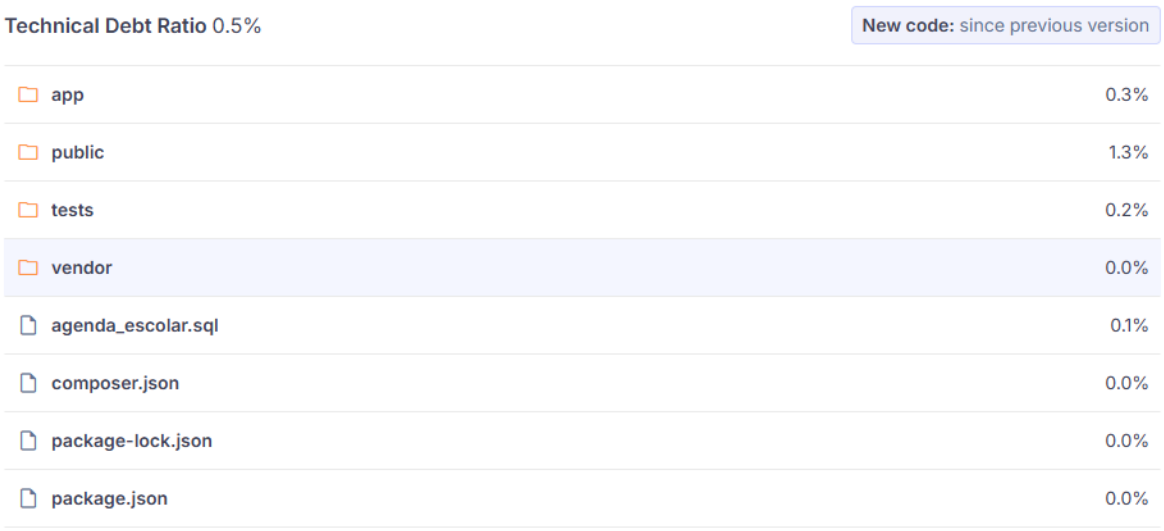
app	22
public	113
tests	22
vendor	0
agenda_escolar.sql	1
composer.json	0
package-lock.json	0
package.json	0

3.3 Deuda técnica

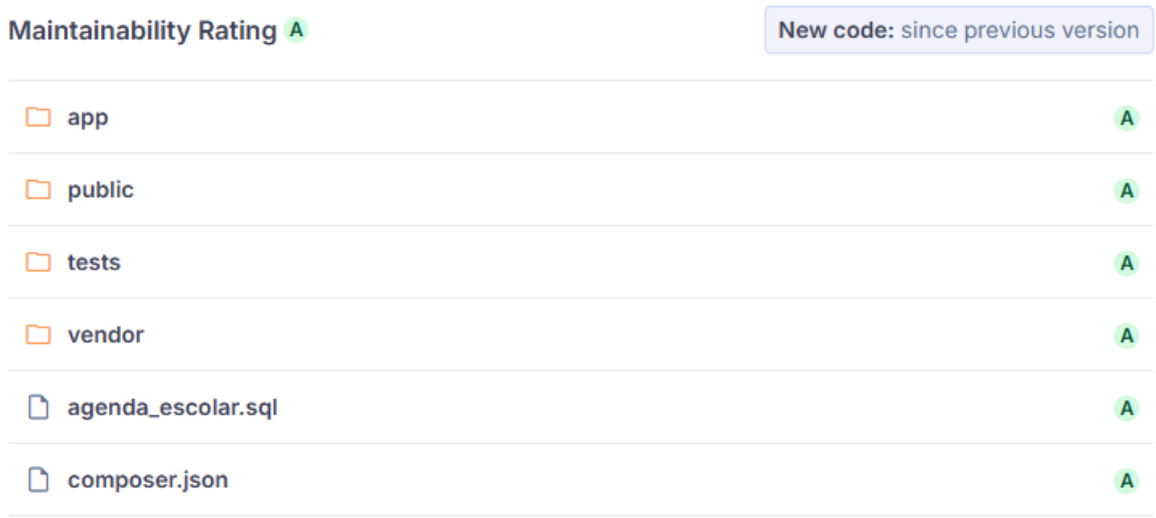
Technical Debt 4d 6h — **New code:** since previous version

app	3h 23min
public	4d 2h
tests	49min
vendor	0
agenda_escolar.sql	4min
composer.json	0
package-lock.json	0
package.json	0

3.4 Radio de deuda técnica

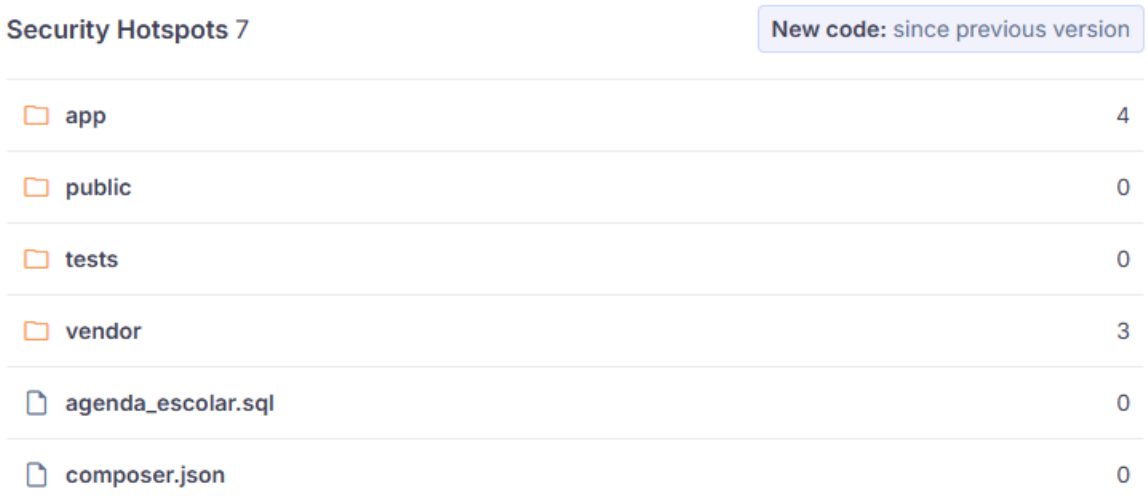


3.5 Rating de mantenibilidad

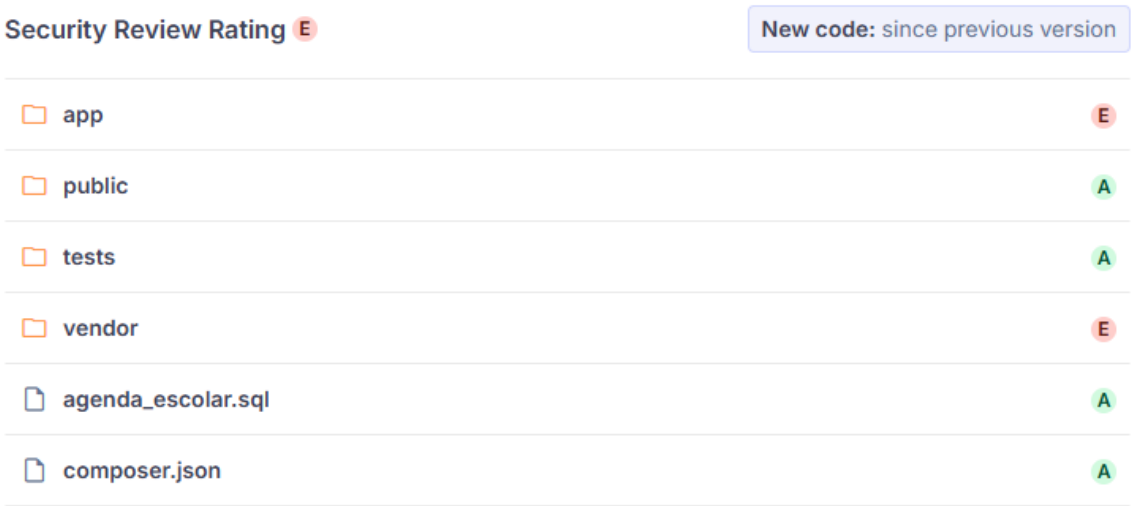


4. Análisis de seguridad (riesgos potenciales)

4.1 Número de hotspots

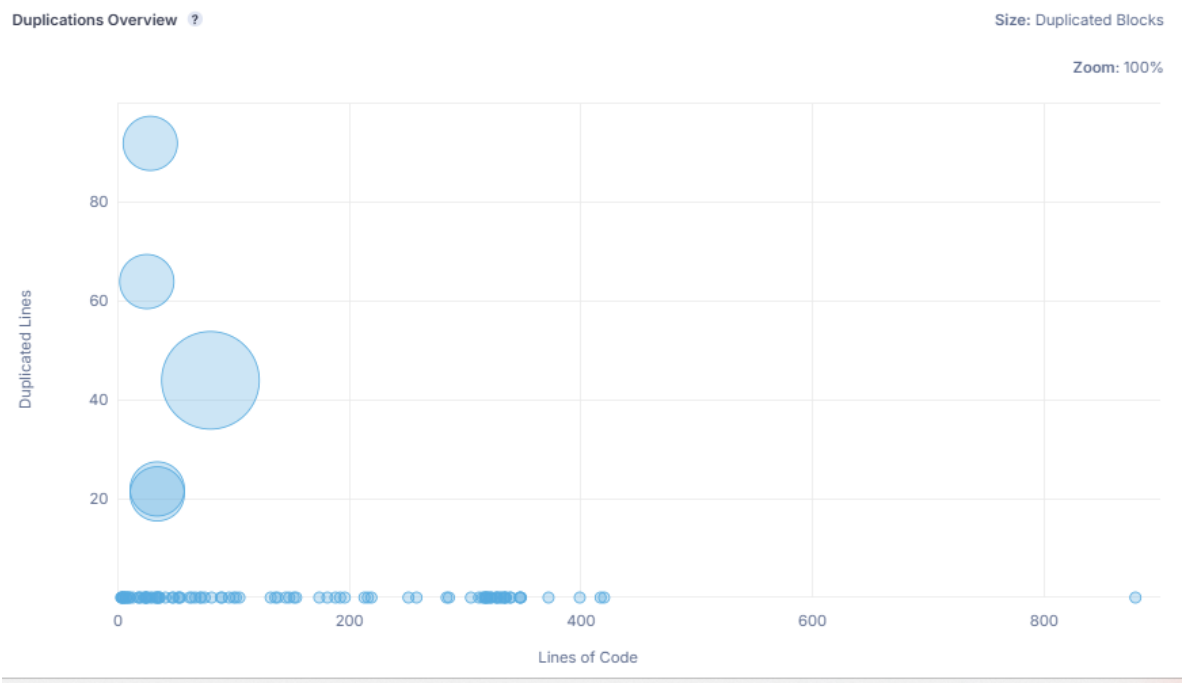


4.2 Rating



5. Duplicación

5.1 Overview



5.2 Líneas duplicadas

Duplicated Lines (%) 0.8%

New code: since previous version

	Duplicated Lines (%)	Duplicated Lines
app	3.9%	156
public	0.0%	0
tests	7.8%	87
vendor	0.0%	0
agenda_escolar.sql	0.0%	0
composer.json	0.0%	0
package-lock.json	0.0%	0
package.json	0.0%	0

5.3 Bloques duplicados

Duplicated Blocks 6 — New code: since previous version

app	2
public	0
tests	4
vendor	0
agenda_escolar.sql	0
composer.json	0
package-lock.json	0
package.json	0

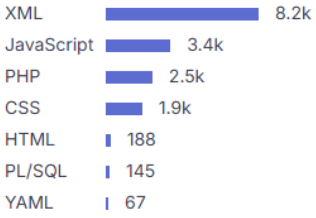
5.4 Archivos duplicados

Duplicated Files 5 — New code: since previous version

app	2
public	0
tests	3
vendor	0
agenda_escolar.sql	0
composer.json	0
package-lock.json	0
package.json	0

6. Tamaño

6.1 Líneas de código



app	1,959
public	5,145
tests	888
vendor	8,301
agenda_escolar.sql	145
composer.json	–

6.2 Líneas totales

Lines 28,790 — **New code:** since previous version

app	3,953
public	6,971
tests	1,119
vendor	11,423
agenda_escolar.sql	285
composer.json	23
package-lock.json	4,993
package.json	23

6.3 Número de sentencias ejecutables

Statements 3,393	**New code:** since previous version
app	972
public	1,975
tests	446
vendor	–
agenda_escolar.sql	0
composer.json	–
package-lock.json	–
package.json	–

6.4 Número de funciones

Functions 472	**New code:** since previous version
app	107
public	249
tests	116
vendor	–
agenda_escolar.sql	0
composer.json	–
package-lock.json	–
package.json	–

ProyConstruccionSoftware > Code > app | View as Tree | 6 files

Functions 107 | **New code:** since previous version

Config	0
Controllers	41
Core	13
Exceptions	0
Models	53
Views	0

6 of 6 shown

ProyConstruccionSoftware > Code > public | View as Tree | 5 files

Functions 249 | **New code:** since previous version

assets	107
js	141
partials	–
styles	–
index.php	1

6.5 Número de clases

ProyConstruccionSoftware > Code | View as Tree | 8 files

Classes 22 | **New code:** since previous version

app	18
public	0
tests	4
vendor	–
agenda_escolar.sql	–
composer.json	–
package-lock.json	–

ProyConstruccionSoftware > Code > app

View as Tree

6 files

Classes 18

New code: since previous version

Config	0
Controllers	7
Core	3
Exceptions	2
Models	6
Views	0

6.6 Número de archivos

Files 131

New code: since previous version

app	22
public	32
tests	15
vendor	58
agenda_escolar.sql	1
composer.json	1
package-lock.json	1
package.json	1

ProyConstruccionSoftware > Code > app

View as Tree

6 files

Files 22

New code: since previous version

Config	1
Controllers	7
Core	3
Exceptions	2
Models	4
Views	5

ProyConstruccionSoftware > Code > public

View as Tree

5 files

Files 32

New code: since previous version

assets	13
js	5
partials	8
styles	5
index.php	1

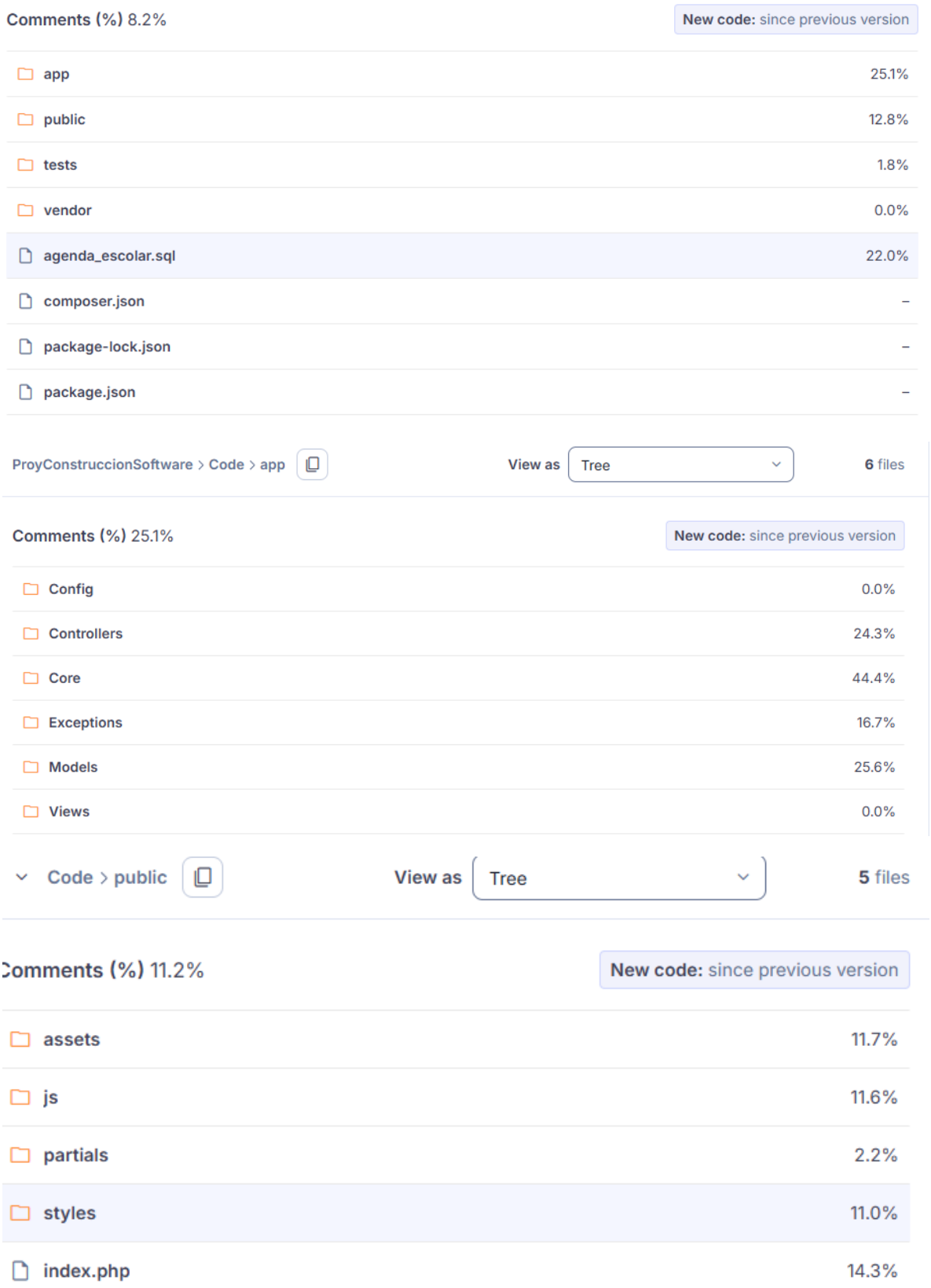
6.7 Líneas de comentarios

Comment Lines 1,468

New code: since previous version

app	657
public	753
tests	16
vendor	1
agenda_escolar.sql	41
composer.json	–
package-lock.json	–
package.json	–

6.8 Porcentaje de líneas comentadas



ProyConstruccionSoftware > Code > public

View as Tree

5 files

Comments (%) 12.8%

New code: since previous version

assets	13.8%
js	12.8%
partials	7.4%
styles	11.0%
index.php	14.3%

7. Complejidad ciclomática

Cyclomatic Complexity 1,461

New code: since previous version

app	309
public	1,032
tests	120
vendor	–
agenda_escolar.sql	0
composer.json	–
package-lock.json	–

Cyclomatic Complexity 309

New code: since previous version

Config	0
Controllers	154
Core	22
Exceptions	0
Models	113
Views	20

Cyclomatic Complexity 1,032 — **New code:** since previous version

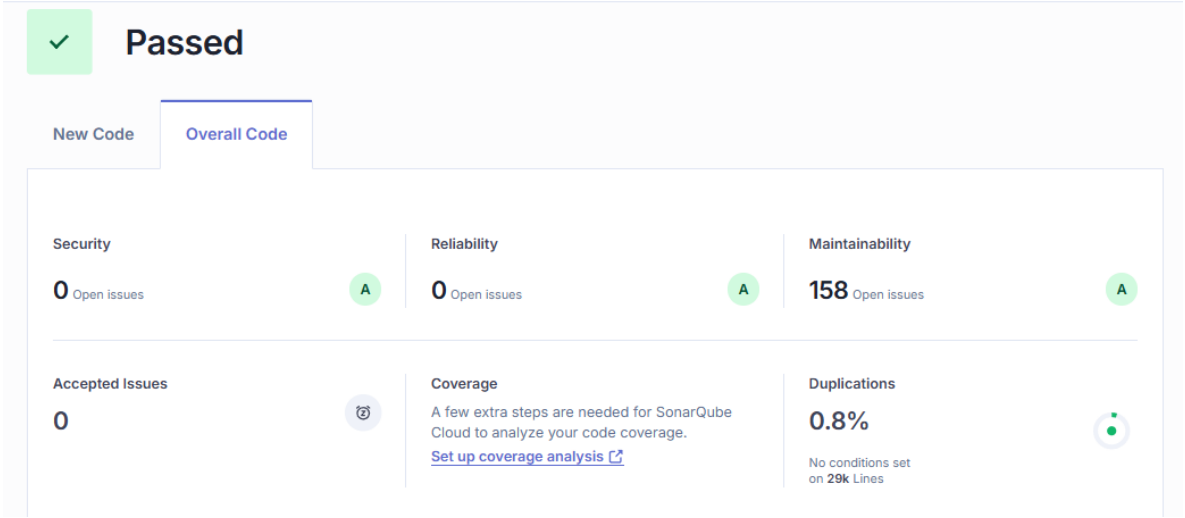
assets	468
js	554
partials	8
styles	–
index.php	2

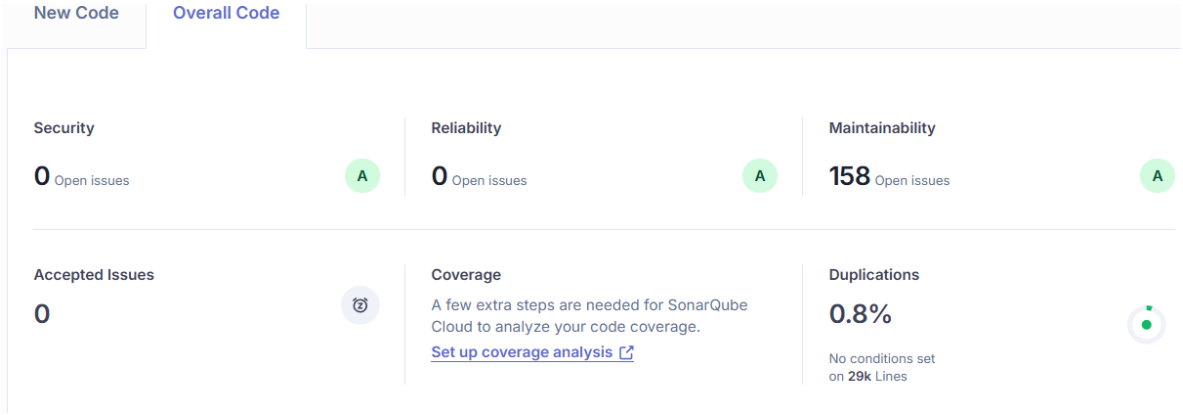
8. Complejidad cognitiva

Cognitive Complexity 1,363 — **New code:** since previous version

app	329
public	1,028
tests	6
vendor	–
agenda_escolar.sql	0
composer.json	0
package-lock.json	0

9. Overall





El análisis global del proyecto en SonarCloud muestra que el código posee buenos niveles de calidad, con indicadores positivos en seguridad, confiabilidad y mantenibilidad. No se detectan vulnerabilidades críticas y el número de bugs es mínimo, lo que refleja un sistema estable y seguro. La deuda técnica es baja y los code smells presentes corresponden principalmente a detalles menores, fáciles de corregir. Además, la duplicación de código es reducida, la complejidad ciclomática está bajo control y la estructura general del proyecto demuestra una arquitectura ordenada y fácil de mantener.

En conjunto, SonarCloud indica que el proyecto es confiable, seguro y técnicamente sólido, con una base adecuada para su crecimiento y mantenimiento futuro.

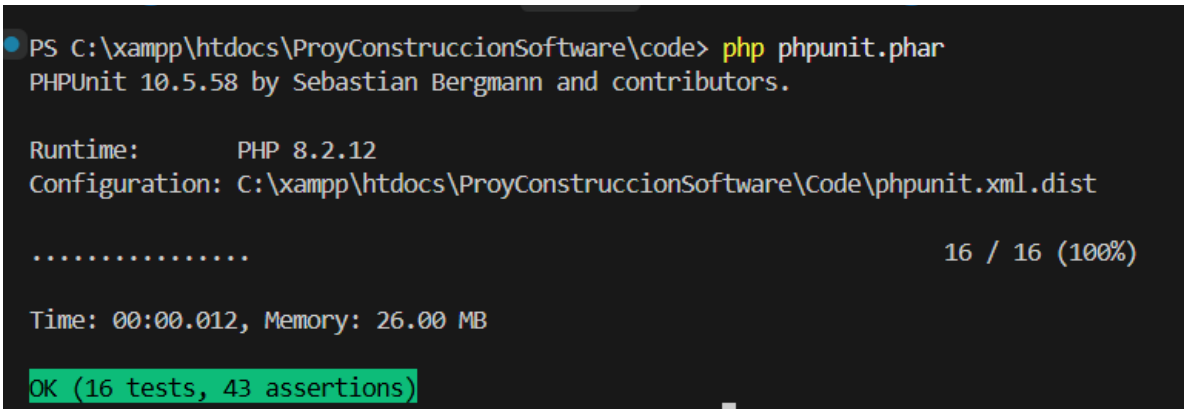
Evidencias de pruebas unitarias

Javascript

```
PS C:\xampp\htdocs\ProyConstruccionSoftware\code> npm test

> code@1.0.0 test
> jest

 PASS  tests/js/mis-calificaciones-detalle.test.js
 PASS  tests/js/mis-calificaciones.test.js
 PASS  tests/js/dashboard.test.js
 PASS  tests/js/sidebar.test.js
 PASS  tests/js/mis-materias.test.js
 PASS  tests/js/modal-filtro.test.js
 PASS  tests/js/modal-nueva-materia.test.js
  ● Console

    console.log
      Inicializando modal materia: listener attached

      at log (public/assets/js/modal-nueva-materia.js:73:11)

    console.log
      Inicializando modal materia: listener attached

      at log (public/assets/js/modal-nueva-materia.js:73:11)

 PASS  tests/js/modal-nueva.test.js
 PASS  tests/js/login.test.js
 PASS  tests/js/ui-helpers.test.js
 PASS  tests/js/buttons.test.js

Test Suites: 11 passed, 11 total
Tests:       32 passed, 32 total
Snapshots:   0 total
Time:        2.084 s
Ran all test suites.
```

Las pruebas unitarias para el frontend se realizaron utilizando la herramienta Jest.

PHP (Backend)



Las pruebas unitarias para el backend se realizaron utilizando la herramienta PHPUnit.