AVANCE PROYECTO FINAL SISTEMA “NEOCDT”

INTEGRANTES:

- SAMUEL SALAZAR TRUJILLO – 2232061
- ROIMAN URREGO ZUÑIGA – 2231385
- ISABELA CABEZAS OBREGÓN – 2226013
- ISABELLA ORTÍZ HERNANDEZ – 2226215
- SEBASTIAN BUSTAMANTE LOPEZ – 2230843

DOCENTE:

RODRIGO ESCOBAR LOPEZ

ASIGNATURA:

INGENIERIA DE SOFTWARE 2

UNIVERSIDAD AUTONOMA DE OCCIDENTE
PERIODO 2025 – 2
CALI - VALLE

CONTEXTO

NeoBank, como banco 100% digital en LATAM, busca optimizar la experiencia de sus clientes mediante el módulo NeoCDT**,** orientado a la apertura y gestión de Certificados de Depósito a Término de manera totalmente digital. Actualmente existen problemas de fricción en el registro, falta de visibilidad en el estado de las solicitudes, operaciones manuales y debilidades en seguridad y calidad del software.

Frente a esta situación, se identificaron los requisitos funcionales y no funcionales que el sistema debe cumplir, priorizados con la técnica MoSCoW**,** con el fin de asegurar un desarrollo alineado a los objetivos del negocio y a las necesidades de los usuarios.

REQUISITOS FUNCIONALES

ID	Requisito	Impacto en la solución	Objetivo asociado	Moscow
RF1	El sistema debe permitir abrir CDTs de forma totalmente digital, sin necesidad de trámites presenciales.	Facilita la inclusión financiera y atrae clientes que buscan procesos ágiles.	Flujo centralizado para solicitudes de CDT.	Must Have
RF2	El sistema debe reutilizar la información ya registrada del cliente (KYC), evitando solicitarla de nuevo.	Reduce fricción, errores y mejora la experiencia de uso.	Reducción de fricción en login y registro.	Must Have
RF3	El sistema debe mostrar al cliente, en tiempo real, el estado de su solicitud (borrador, en validación, aprobada, rechazada, cancelada).	Aporta transparencia, confianza y control al cliente.	Transparencia de estados y trazabilidad.	Must Have
RF4	El sistema debe permitir al cliente renovar su CDT de forma automática o manual según prefiera.	Favorece la fidelización y retención de clientes.	Flujo centralizado y trazabilidad.	Should Have
RF5	El sistema debe ofrecer un panel para que los agentes bancarios gestionen y supervisen todas las solicitudes.	Disminuye la operación manual y mejora la eficiencia interna.	Flujo centralizado y operación eficiente.	Must Have

RF6	El sistema debe enviar notificaciones automáticas (correo, SMS o app) al cliente cuando cambie el estado de un CDT.	Mantiene informado al cliente en todo momento.	Transparencia de estados y trazabilidad.	Must Have
RF7	El sistema debe permitir al cliente cancelar su solicitud de CDT mientras esté en estado de borrador o validación.	Da flexibilidad y mayor control sobre sus decisiones.	Flujo centralizado y experiencia de usuario.	Should Have
RF8	El sistema debe incluir autenticación segura para confirmar apertura, cancelación o renovación de un CDT.	Refuerza seguridad y confianza en operaciones sensibles.	Reducción de fricción con seguridad.	Must Have
RF9	El sistema debe permitir a los administradores consultar reportes y métricas sobre CDTs (aperturas, renovaciones, cancelaciones).	Mejora la toma de decisiones y el control de calidad.	Métricas y calidad del sistema.	Could Have

REQUISITOS NO FUNCIONALES

ID	Requisito	Categoría	Especificación (cómo se mide / aplica)	Prioridad (MoSCoW)	Justificación

RNF1	El sistema debe encriptar toda la información del cliente, tanto en almacenamiento como en transmisión.	**Seguridad**	Uso obligatorio de **TLS 1.3** en transmisión y **AES-256** en la base de datos.	Must Have	Es obligatorio en un banco digital para proteger datos sensibles y cumplir la ley.
RNF2	El sistema debe implementar autenticación multifactor (MFA) en operaciones sensibles.	**Seguridad**	Confirmación de operaciones mediante **OTP, biometría o token digital**.	Must Have	Evita fraudes y accesos indebidos.
RNF3	El sistema debe garantizar la disponibilidad del servicio de manera continua.	**Disponibilidad**	Uptime mínimo del **99.5% mensual** con redundancia en servidores.	Must Have	Los clientes esperan acceder a su dinero en todo momento.
RNF4	El sistema debe responder en tiempos adecuados a las acciones del cliente.	**Rendimiento**	Consultas y aperturas de CDTs deben resolverse en **máx. 3 segundos**.	Should Have	Mejora la experiencia del usuario, aunque puede optimizarse con el tiempo.
RNF5	El sistema debe soportar picos de uso altos sin degradar su rendimiento.	**Escalabilidad**	Manejo de al menos **10.000 usuarios concurrentes**.	Should Have	Garantiza estabilidad en campañas o temporadas de alta demanda.
RNF6	El sistema debe contar con pruebas automatizadas de calidad integradas al pipeline.	**Calidad del software**	Cobertura mínima del **80% en pruebas unitarias** dentro de CI/CD.	Must Have	Reduce errores en producción y asegura confiabilidad.
RNF7	El sistema debe registrar y auditar todas	**Seguridad / Auditoría**	Guardar usuario, fecha, acción y	Must Have	Necesario para cumplir regulaciones

	las operaciones realizadas por los usuarios.		resultado en un **log seguro** durante mínimo 5 años.		y auditorías financieras.

HISTORIAS DE USUARIO

1. **Feature:** Abrir un CDT
   Como cliente ***quiero*** poder abrir un CDT de manera digital ***para*** evitar largas filas y tiempos de espera.
   ❖ **Scenario:** Apertura exitosa de un CDT
   Given el cliente inicia sección en el sistema
   And encuentra el módulo “abrir nuevo CDT”
   When completa los datos requeridos
   Then el sistema registra la solicitud en estado “Pendiente por aprobación”
   And el sistema muestra el mensaje “Solicitud de apertura de CDT exitoso”

2. **Feature:** Registro de cliente nuevo
   Como cliente nuevo ***quiero*** poder registrar mis datos a una cuenta ***para*** gestionar nuevas solicitudes de CDT.
   ❖ **Scenario:** Registro exitoso
   Given el cliente quiere abrir un CDT
   And no tiene una cuenta vigente en el sistema
   When ingresa todos los datos personales requeridos para el registro
   Then el sistema crea una cuenta de usuario
   And el usuario recibe un correo de confirmación de apertura de cuenta

3. **Feature:** Uso de datos registrados previamente
   Como cliente ***quiero*** utilizar los datos ya registrados en el sistema sin la necesidad de repetir estos al momento de acceder o abrir un CDT, ***para*** evitar réplicas de información y realizar pasos innecesarios.
   ❖ **Scenario:** Reutilizacion de datos ya registrados
   Given el cliente inicia sesión en el sistema
   When el cliente da click en “acceder a mi CDT”
   Then el sistema carga automáticamente los datos ya registrados en el perfil del usuario

4. **Feature:** Estado de las solicitudes
   Como cliente ***quiero*** tener claridad sobre el estado de mi solicitud del CDT ***para*** poder tener un mayor entendimiento y transparencia en el proceso.
   ❖ **Scenario:** Visualización del estado de la solicitud
   Given el cliente inicia sesión en su cuenta
   When da click en el apartado de “Mis solicitudes”
   Then el sistema muestra una lista con las solicitudes del cliente junto con su estado actual (Pendiente, Aprobado, Rechazado, Cancelado).

5. **Feature:** Renovación de CDTs
   Como cliente ***quiero*** poder renovar mi CDT de forma manual o automática según mis preferencias, ***para*** asegurar mis inversiones.
   - **Scenario:** Renovación automatizada

   Given el cliente tiene un CDT con fecha próxima a vencer
   And tiene activa la opción de "renovar automáticamente"
   When llega la fecha de vencimiento del CDT
   Then el sistema genera una nueva solicitud de CDT con los datos previamente registrados.

6. **Feature:** Panel de Visualización de agente
   Como agente ***quiero*** tener un panel centralizado donde tenga acceso a la visualización completa de todas las solicitudes, ***para*** así reducir procesos manuales.
   - **Scenario:** Acceso al panel de visualizaciones

   Given el agente inicia sesión en el sistema
   When el agente accede al panel
   Then el sistema despliega una lista de todas las solicitudes de CDT

7. **Feature:** Notificaciones de estado de CDT
   Como cliente ***quiero*** recibir notificaciones avisándome del cambio de estado en mi solicitud de CDT, **para** así estar al tanto por si se presenta algún inconveniente.
   - **Scenario:** Notificaciones de cambio en el estado de un CDT

   Given el cliente tiene una solicitud de CDT en proceso
   When la solicitud de CDT cambie de estado (Aprobado, Rechazado, Cancelado)
   Then el sistema envía al correo del cliente la notificación de este cambio.

8. **Feature:** Cancelación de solicitud de CDT
   Como cliente ***quiero*** poder cancelar mi solitud de CDT mientras esta esté en estado de borrado o validación ***para*** tener control sobre mis decisiones.
   - **Scenario:** Cancelación de solicitud pendiente por aprobación

   Given el cliente tiene una solicitud en estado "Pendiente por aprobación"
   When hace click en la opción "Cancelar Solicitud"
   Then el sistema debe cambiar el estado de la solicitud a "cancelado"

9. **Feature:** Proceso de Autenticación
   Como cliente ***quiero*** confirmar mis solicitudes a través de un proceso de autenticación ***para*** así asegurar la confidencialidad en mis datos y seguridad en mis solicitudes.
   - **Scenario:** Confirmación de solicitud

   Given el cliente tiene una solicitud pendiente de aprobación
   When el sistema solicita una llave de acceso o un token
   Then el cliente ingresa el token correspondiente
   And la solicitud es confirmada

10. **Feature:** Consulta de reportes
    Como administrador ***quiero*** poder consultar los reportes y las métricas de las solicitudes de CDT activas ***para*** así tener un análisis completo del producto y tomar decisiones.
    ❖ **Scenario:** Consulta de reportes de CDTs
    Given el administrador inicia sesión en el sistema
    And accede al panel de control
    When selecciona la opción de “ver reportes”
    Then el sistema despliega estadísticas de CDTs pendientes por aprobación, activos, renovados y cancelados.

ATRIBUTOS – ESCENARIOS DE CALIDAD

De acuerdo con el contexto, hemos identificado los atributos más relevantes:

1. **Seguridad:** El sistema NeoCDT maneja información muy sensible como lo son datos personales de los clientes y su dinero en los CDT. Si la seguridad es débil, el riesgo no es solo que un atacante robe información o dinero, sino también que el banco enfrente problemas legales, pérdida de reputación y pérdida de confianza de los usuarios. Además, el mismo contexto indica que no existen análisis estáticos ni pruebas de seguridad, lo que significa que las vulnerabilidades podrían llegar fácilmente a producción sin ser detectadas.

2. **Usabilidad:** El primer contacto de los usuarios con el sistema es el proceso de apertura de un CDT. En el contexto se dice que actualmente es un proceso engorroso, pues los clientes deben repetir información ya que el sistema no reconoce cuando ya hay datos previos y el flujo es poco claro. Esto provoca abandono (clientes que nunca terminan el registro), genera frustración y aumenta la carga manual de los agentes para corregir o validar solicitudes.

3. **Testabilidad:** El contexto indica que no existen pruebas automatizadas ni análisis estático del código. Eso significa que cada vez que los desarrolladores hacen un cambio, no hay manera confiable de saber si rompieron algo que antes funcionaba. Esto lleva a que los errores lleguen hasta producción, lo que obliga a los agentes a arreglarlos manualmente y afecta la experiencia del cliente. Además, sin pruebas es difícil mantener y evolucionar el sistema con seguridad.

Escenario de calidad – Seguridad - Detección y bloqueo de accesos / operaciones no autorizadas

Fuente: Atacante externo o credenciales comprometidas.

Un atacante o alguien con credenciales robadas intenta acceder al sistema para abrir o modificar un CDT usando esas credenciales, posiblemente desde una ubicación o dispositivo inusual.

Estímulo: Intento de abrir o manipular un CDT usando credenciales robadas o solicitudes previamente anuladas.

El usuario (o atacante) enciende una sesión en la app o web y pulsa “Crear CDT” o “Modificar solicitud”, pero lo hace desde un dispositivo desconocido o IP geográficamente distinta a su comportamiento normal.

Artefacto: Servicio de autenticación, API de CDT, base de datos KYC (Know Your Costumer) y logs de auditoría.

El frontend envía la petición al servicio de autenticación y a la API de gestión de CDT. Posteriormente estos servicios consultan la base de datos KYC y generan eventos en los logs/auditoría.

Ambiente: Producción / uso real.

La situación ocurre en la versión en vivo de la app con usuarios reales, durante un horario normal con carga moderada. No es un pico extremo ni una caída de red masiva.

Respuesta: El sistema identifica la actividad sospechosa, bloquea la operación y pide verificación adicional; registra el evento y alerta al equipo de seguridad.

Se detecta que la sesión es anómala, la operación queda bloqueada, se muestra al usuario un mensaje claro (por ejemplo: “Actividad sospechosa: verifique por SMS/2FA”), se solicita un paso extra de verificación y el intento queda registrado en el log de auditoría; si no se verifica, la acción no se completa.

Medida de respuesta: ≥ 99.9% de intentos maliciosos detectados y bloqueados; tiempo medio de detección < 5 minutos; 100% de cuentas administrativas protegidas por 2FA.

Métrica de éxito:

- Detección y bloqueo efectivo en ≥ 99.9% de intentos maliciosos durante un periodo representativo (por ejemplo: 30 días).
- Tiempo medio de detección < 5 minutos.
- 100% de cuentas administrativas con 2FA activo.
- 0 vulnerabilidades críticas abiertas por despliegue (según SAST).

Estas métricas se revisan periódicamente (mensual/trimestral) para validar cumplimiento.

Escenario de calidad – Usabilidad – Registros reutilizables sin repetición de datos

Fuente: Usuario nuevo o existente que abre un CDT.

Un cliente quiere abrir un CDT desde la app o la web. Puede ser un usuario nuevo o uno que ya tiene datos en el sistema.

Estímulo: Inicia el proceso de apertura y presiona “Enviar” después de completar sus datos y subir documentos.

El usuario completa el formulario y pulsa “Enviar” para iniciar la verificación KYC (Know Your Costumer) y la creación de la solicitud de CDT.

Artefacto: Frontend (formularios), servicio KYC, repositorio/tabla de clientes y UI de estados.

El formulario envía los datos al servicio KYC, que consulta la base de datos de clientes para identificar duplicados y guarda la solicitud con su estado inicial.

Ambiente: App en condiciones normales de red.

La acción ocurre en uso diario, con concurrencia moderada y sin condiciones de pico extremo.

Respuesta: El sistema reutiliza datos existentes, guía el flujo y confirma la creación sin pedir repetir información innecesaria.

Si el usuario ya existe, el sistema no solicita de nuevo datos básicos; muestra pasos claros y el estado actual (por ejemplo: “Borrador → En validación → Aprobado”). Después de presionar “Enviar” el usuario ve una confirmación clara y la nueva solicitud aparece en su historial.

Medida de respuesta:

- Tasa de éxito end-to-end del registro ≥ 95%.
- Tiempo para completar registro (P95) ≤ 4 minutos.
- Tasa de abandono en registro < 5%.
- KYC duplicados en la base de datos < 1% (o reducción ≥ 90% respecto a estado previo).

Métrica de éxito:

- ≥ 95% de usuarios que inician el registro completan el proceso en el periodo de referencia (por ejemplo: 30 días).
- El 95º percentil del tiempo de completitud ≤ 4 minutos.
- Tasa de abandono durante el paso KYC < 5% por cohorte semanal.
- Porcentaje de clientes con registros duplicados < 1% en la base de datos (o reducción ≥ 90% respecto a línea base), medido mensualmente.

Escenario de calidad – Testabilidad – Cambios seguros sin romper el código

Fuente: Desarrollador que hace un cambio en el código.

Un desarrollador sube un cambio que modifica cómo se valida la información del cliente o cómo se crea una solicitud de CDT.

Estímulo: Pide fusionar (merge) ese cambio a la rama principal.

Al crear la solicitud de fusión, el sistema automático corre varias pruebas y chequeos antes de permitir que el cambio avance.

Artefacto: Repositorio de código y pipeline automático (pruebas y análisis).

El cambio pasa por pruebas rápidas (unitarias), pruebas más completas y un análisis de seguridad automático antes de poder integrarse.

Ambiente: Entorno de integración y staging (no se prueba en usuarios reales).

Las pruebas corren en un entorno controlado que imita al de producción, por lo que no se usan datos reales ni se afectan usuarios.

Respuesta: Si alguna prueba o el análisis de seguridad falla, el merge se bloquea y el cambio NO llega a producción; si todo pasa, el cambio se puede promocionar para pruebas finales y luego desplegar.

Sólo se permite avanzar cambios que no rompan funciones existentes y que no introduzcan problemas de seguridad.

Medida de respuesta:

- Cobertura mínima en código crítico (validaciones KYC/CDT) ≥ 80%.
- 0 vulnerabilidades críticas permitidas en cada PR; vulnerabilidades altas deben corregirse antes del merge.
- ≥ 95% de PRs con pipeline exitoso.
- Tiempo promedio para arreglar un bug crítico en staging < 24 horas.
- Tasa de rollback en producción ≤ 1% de despliegues al mes.

Métrica de éxito:

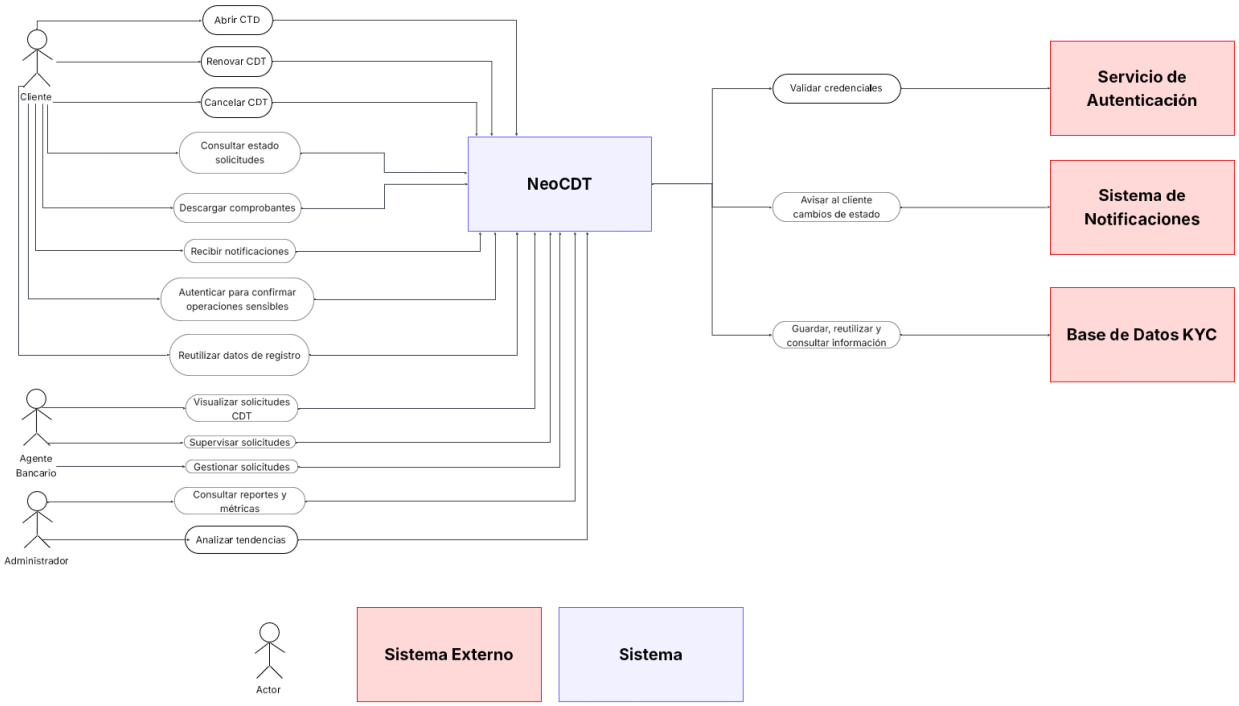
Que se cumplan las cinco medidas anteriores durante el periodo de referencia (por ejemplo: 30 – 90 días):

- cobertura ≥ 80%,
- 0 críticas en PRs,
- ≥ 95% builds verdes,
- MTTR < 24 h,
- rollbacks ≤ 1% mensual.

DISEÑO ARQUITECTONICO

Vistas Arquitectónicas

1. Vista de Contexto



La Vista de Contexto del sistema NeoCDT muestra cómo este se relaciona tanto con los actores humanos como con los sistemas externos necesarios para su funcionamiento.

En primer lugar, el Cliente es el actor principal, ya que interactúa con NeoCDT para abrir, renovar, cancelar y consultar Certificados de Depósito a Término (CDTs). Además, puede descargar comprobantes, recibir notificaciones y reutilizar sus datos previamente registrados para evitar reprocesos.

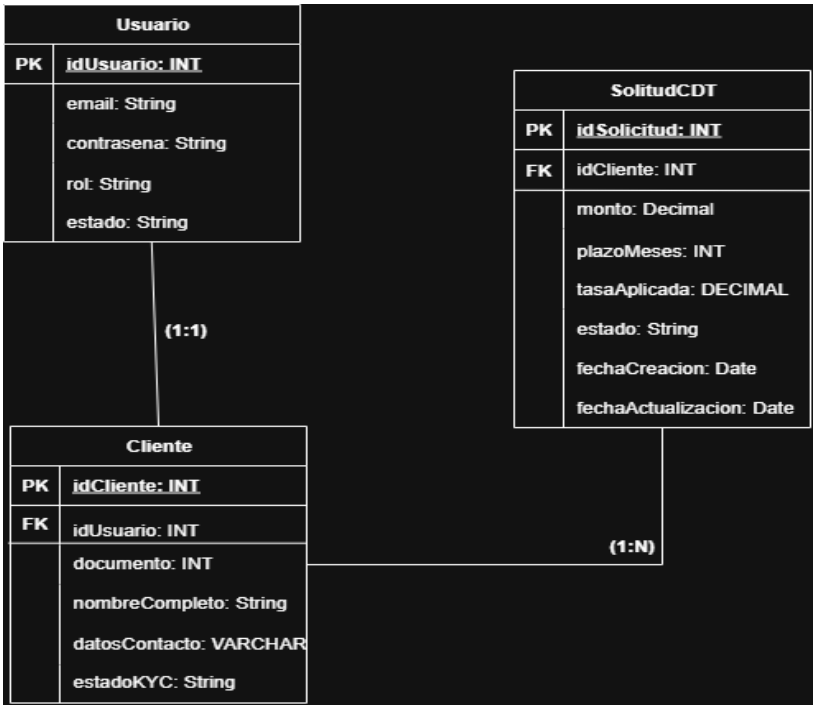
El Agente Bancario utiliza un panel centralizado que le permite visualizar, supervisar y gestionar las solicitudes de CDT, lo que reduce la carga de trabajo manual y mejora la eficiencia operativa. Por su parte, el Administrador consulta reportes y métricas del sistema, con el fin de analizar tendencias y tomar decisiones estratégicas basadas en datos.

El sistema NeoCDT se apoya en tres componentes externos clave:

- El Servicio de Autenticación, encargado de validar credenciales y reforzar la seguridad de operaciones sensibles mediante autenticación multifactor.
- La Base de Datos KYC, donde se guarda, reutiliza y consulta la información de los clientes y de las solicitudes de CDT, garantizando trazabilidad y consistencia en los datos.
- El Sistema de Notificaciones, que se encarga de mantener informado al cliente mediante alertas sobre el estado de sus solicitudes (por correo, SMS o notificaciones push).

De esta forma, el diagrama refleja cómo NeoCDT centraliza la gestión de CDTs, a la vez que se conecta con usuarios y sistemas externos para ofrecer seguridad, transparencia y eficiencia en la operación.

2. Vista de Información

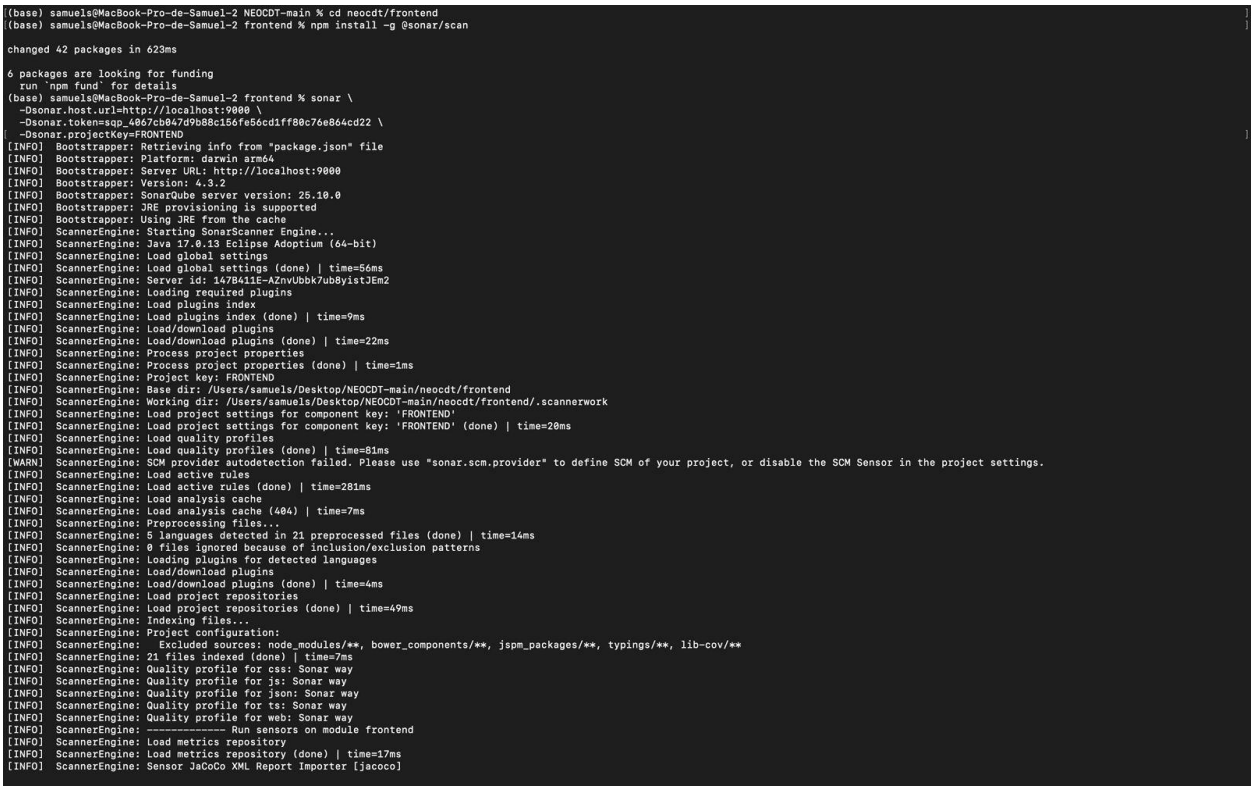


La Vista de Información del sistema NeoCDT se centra en tres entidades principales: Usuario, Cliente y SolicitudCDT. La entidad Usuario gestiona las credenciales de acceso y el rol dentro del sistema (cliente, agente o administrador). Cada Usuario se asocia de manera uno a uno con un Cliente, lo que significa que cada cliente registrado cuenta con un único perfil de acceso al sistema. La entidad Cliente contiene la información personal y regulatoria, incluyendo documento, nombre, datos de contacto y el estado KYC (*Know Your Customer*), que corresponde al proceso obligatorio en el sector financiero para verificar la identidad del cliente y cumplir con normativas de seguridad y prevención de fraude. A su vez, la entidad Cliente se relaciona de forma uno a muchos con SolicitudCDT, ya que un cliente puede generar varias solicitudes, pero cada solicitud pertenece

únicamente a un cliente. La entidad SolicitudCDT registra la operación de apertura, renovación o cancelación de un certificado, con atributos como monto, plazo, tasa aplicada, estado, canal y fechas de creación y actualización. Estas relaciones aseguran consistencia y trazabilidad, ya que permiten vincular la autenticación de los usuarios con sus datos personales y con las solicitudes de productos financieros, garantizando un flujo centralizado y seguro en la gestión de CDTs.

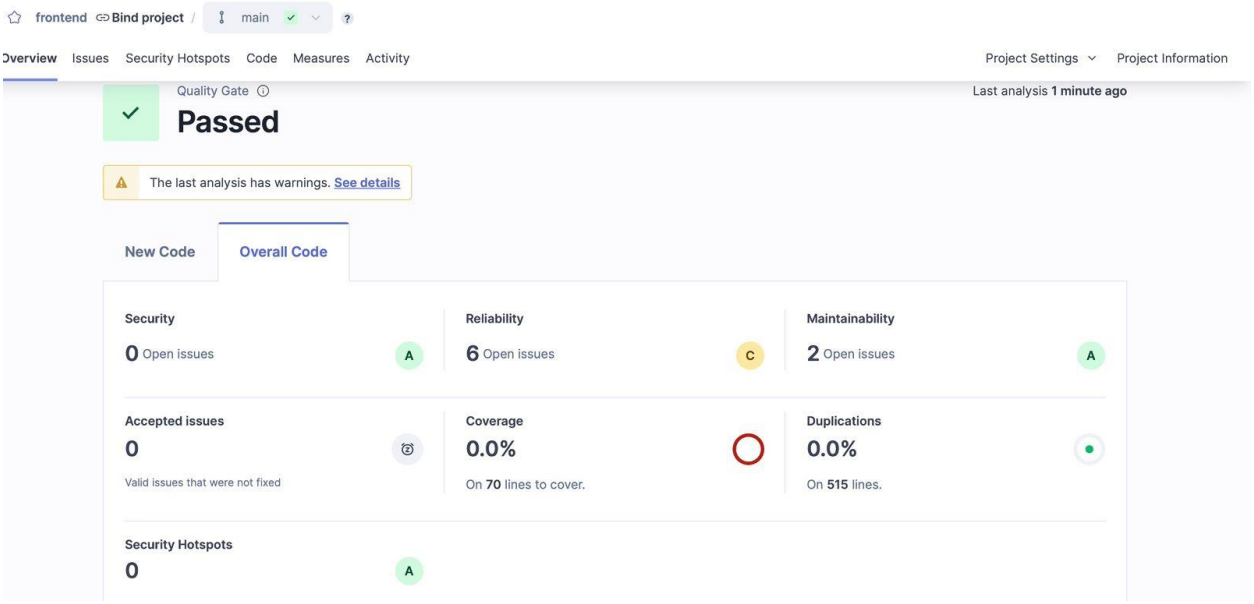
Pruebas SonarQube

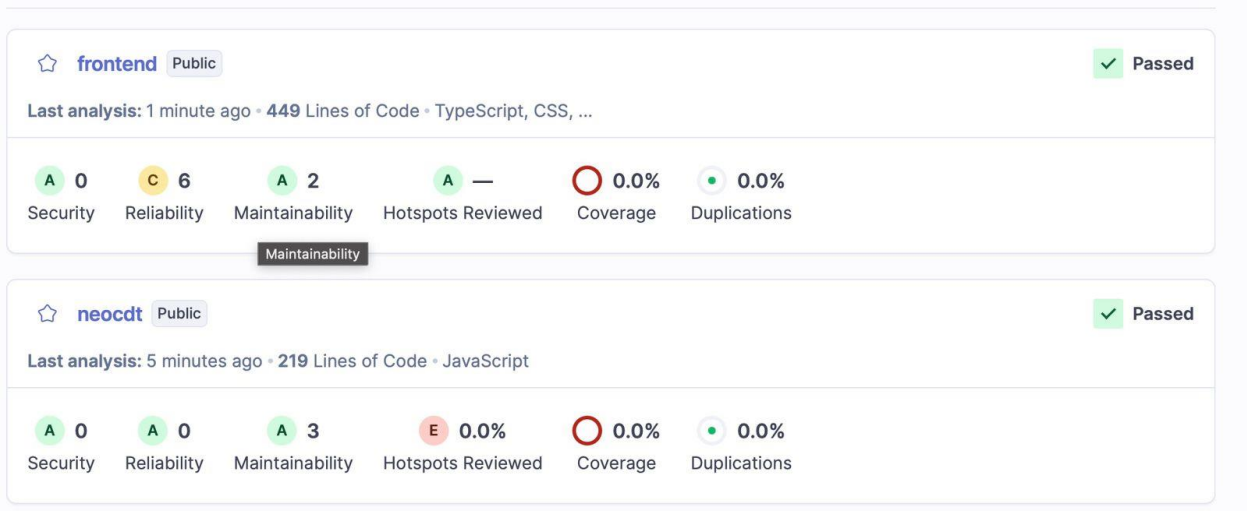
Fronted



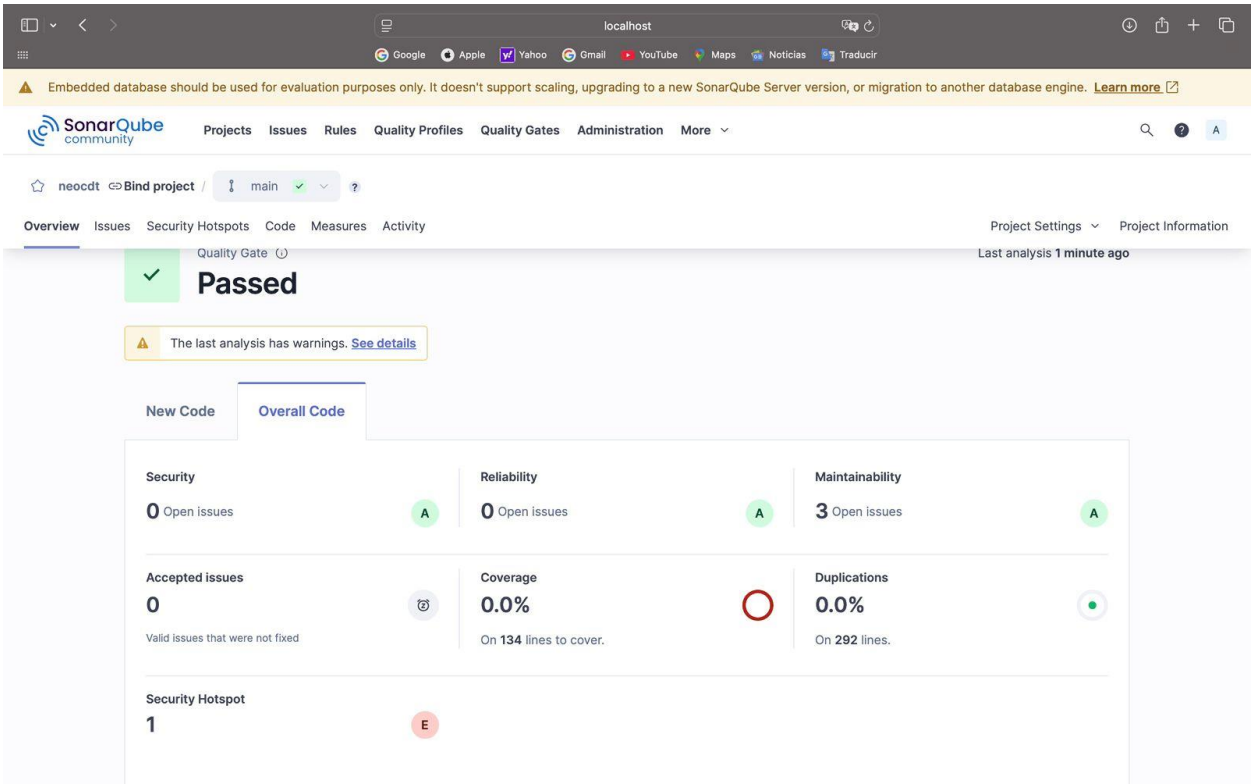
```
(base) samuels@MacBook-Pro-de-Samuel-2 NEOCDT-main % cd neocdt/frontend
(base) samuels@MacBook-Pro-de-Samuel-2 frontend % npm install -g @sonar/scan

changed 42 packages in 623ms

6 packages are looking for funding
  run `npm fund` for details
(base) samuels@MacBook-Pro-de-Samuel-2 frontend % sonar \
  -Dsonar.host.url=http://localhost:9000 \
  -Dsonar.token=sqp_4067cb047d9b88c156fe56cd1ff80c76e864cd22 \
  -Dsonar.projectKey=FRONTEND
[INFO]  Bootstrapper: Retrieving info from "package.json" file
[INFO]  Bootstrapper: Platform: darwin arm64
[INFO]  Bootstrapper: Server URL: http://localhost:9000
[INFO]  Bootstrapper: Version: 4.3.2
[INFO]  Bootstrapper: SonarQube server version: 25.10.0
[INFO]  Bootstrapper: JRE provisioning is supported
[INFO]  Bootstrapper: Using JRE from the cache
[INFO]  ScannerEngine: Starting SonarScanner Engine...
[INFO]  ScannerEngine: Java 17.0.13 Eclipse Adoptium (64-bit)
[INFO]  ScannerEngine: Load global settings
[INFO]  ScannerEngine: Load global settings (done) | time=56ms
[INFO]  ScannerEngine: Server id: 1478411E-AZnvUbbk7ub8yistJEm2
[INFO]  ScannerEngine: Loading required plugins
[INFO]  ScannerEngine: Load plugins index
[INFO]  ScannerEngine: Load plugins index (done) | time=9ms
[INFO]  ScannerEngine: Load/download plugins
[INFO]  ScannerEngine: Load/download plugins (done) | time=22ms
[INFO]  ScannerEngine: Process project properties
[INFO]  ScannerEngine: Process project properties (done) | time=1ms
[INFO]  ScannerEngine: Project key: FRONTEND
[INFO]  ScannerEngine: Base dir: /Users/samuels/Desktop/NEOCDT-main/neocdt/frontend
[INFO]  ScannerEngine: Working dir: /Users/samuels/Desktop/NEOCDT-main/neocdt/frontend/.scannerwork
[INFO]  ScannerEngine: Load project settings for component key: 'FRONTEND'
[INFO]  ScannerEngine: Load project settings for component key: 'FRONTEND' (done) | time=20ms
[INFO]  ScannerEngine: Load quality profiles
[INFO]  ScannerEngine: Load quality profiles (done) | time=81ms
[WARN]  ScannerEngine: SCM provider autodetection failed. Please use "sonar.scm.provider" to define SCM of your project, or disable the SCM Sensor in the project settings.
[INFO]  ScannerEngine: Load active rules
[INFO]  ScannerEngine: Load active rules (done) | time=281ms
[INFO]  ScannerEngine: Load analysis cache
[INFO]  ScannerEngine: Load analysis cache (404) | time=7ms
[INFO]  ScannerEngine: Preprocessing files...
[INFO]  ScannerEngine: 5 languages detected in 21 preprocessed files (done) | time=14ms
[INFO]  ScannerEngine: 0 files ignored because of inclusion/exclusion patterns
[INFO]  ScannerEngine: Loading plugins for detected languages
[INFO]  ScannerEngine: Load/download plugins
[INFO]  ScannerEngine: Load/download plugins (done) | time=4ms
[INFO]  ScannerEngine: Load project repositories
[INFO]  ScannerEngine: Load project repositories (done) | time=49ms
[INFO]  ScannerEngine: Indexing files...
[INFO]  ScannerEngine: Project configuration:
[INFO]  ScannerEngine:   Excluded sources: node_modules/**, bower_components/**, jspm_packages/**, typings/**, lib-cov/**
[INFO]  ScannerEngine: 21 files indexed (done) | time=7ms
[INFO]  ScannerEngine: Quality profile for css: Sonar way
[INFO]  ScannerEngine: Quality profile for js: Sonar way
[INFO]  ScannerEngine: Quality profile for json: Sonar way
[INFO]  ScannerEngine: Quality profile for ts: Sonar way
[INFO]  ScannerEngine: Quality profile for web: Sonar way
[INFO]  ScannerEngine: ------------ Run sensors on module frontend
[INFO]  ScannerEngine: Load metrics repository
[INFO]  ScannerEngine: Load metrics repository (done) | time=17ms
[INFO]  ScannerEngine: Sensor JaCoCo XML Report Importer [jacoco]
```

```
[INFO]  ScannerEngine: WebSocket client connected on /ws
[INFO]  ScannerEngine: Plugin version: [11.4.1.34873]
[INFO]  ScannerEngine: Found 1 tsconfig.json file(s): [/Users/samuels/Desktop/NEOCDT-main/neocdt/frontend/tsconfig.json]
[INFO]  ScannerEngine: 9 source files to be analyzed
[INFO]  ScannerEngine: Creating TypeScript program
[INFO]  ScannerEngine: TypeScript(5.9.2) configuration file /Users/samuels/Desktop/NEOCDT-main/neocdt/frontend/tsconfig.json
[INFO]  ScannerEngine: Creating TypeScript program
[INFO]  ScannerEngine: TypeScript(5.9.2) configuration file /Users/samuels/Desktop/NEOCDT-main/neocdt/frontend/tsconfig.app.json
[INFO]  ScannerEngine: Creating TypeScript program
[INFO]  ScannerEngine: TypeScript(5.9.2) configuration file /Users/samuels/Desktop/NEOCDT-main/neocdt/frontend/tsconfig.node.json
[INFO]  ScannerEngine: Found 1 file(s) not part of any tsconfig.json: they will be analyzed without type information
[INFO]  ScannerEngine: 9/9 source files have been analyzed
[INFO]  ScannerEngine: Hit the cache for 0 out of 9
[INFO]  ScannerEngine: Miss the cache for 9 out of 9: ANALYSIS_MODE_INELIGIBLE [9/9]
[INFO]  ScannerEngine: Sensor JavaScript/TypeScript analysis [javascript] (done) | time=3028ms
[INFO]  ScannerEngine: Sensor JavaScript inside HTML analysis [javascript]
[INFO]  ScannerEngine: 1 source file to be analyzed
[INFO]  ScannerEngine: 1/1 source file has been analyzed
[INFO]  ScannerEngine: Hit the cache for 0 out of 1
[INFO]  ScannerEngine: Miss the cache for 1 out of 1: ANALYSIS_MODE_INELIGIBLE [1/1]
[INFO]  ScannerEngine: Sensor JavaScript inside HTML analysis [javascript] (done) | time=21ms
[INFO]  ScannerEngine: Sensor CSS Rules [javascript]
[INFO]  ScannerEngine: 3 source files to be analyzed
[INFO]  ScannerEngine: 3/3 source files have been analyzed
[INFO]  ScannerEngine: Hit the cache for 0 out of 0
[INFO]  ScannerEngine: Miss the cache for 0 out of 0
[INFO]  ScannerEngine: Sensor CSS Rules [javascript] (done) | time=50ms
[INFO]  ScannerEngine: Sensor CSS Metrics [javascript]
[INFO]  ScannerEngine: Sensor CSS Metrics [javascript] (done) | time=11ms
[INFO]  ScannerEngine: Sensor TextAndSecretsSensor [text]
[INFO]  ScannerEngine: Available processors: 8
[INFO]  ScannerEngine: Using 8 threads for analysis.
[INFO]  ScannerEngine: The property "sonar.tests" is not set. To improve the analysis accuracy, we categorize a file as a test file if any of the following is true:
  * The filename starts with "test"
  * The filename contains "test." or "tests."
  * Any directory in the file path is named: "doc", "docs", "test" or "tests"
  * Any directory in the file path has a name ending in "test" or "tests"

[INFO]  ScannerEngine: Start fetching files for the text and secrets analysis
[INFO]  ScannerEngine: Using Git CLI to retrieve untracked files
[WARN]  ScannerEngine: Retrieving only language associated files, make sure to run the analysis inside a git repository to make use of inclusions specified via "sonar.text.inclusions"
[INFO]  ScannerEngine: Starting the text and secrets analysis
[INFO]  ScannerEngine: 17 source files to be analyzed for the text and secrets analysis
[INFO]  ScannerEngine: 17/17 source files have been analyzed for the text and secrets analysis
[INFO]  ScannerEngine: Sensor TextAndSecretsSensor [text] (done) | time=348ms
[INFO]  ScannerEngine: ------------ Run sensors on project
[INFO]  ScannerEngine: Sensor Zero Coverage Sensor
[INFO]  ScannerEngine: Sensor Zero Coverage Sensor (done) | time=3ms
[INFO]  ScannerEngine: ------------ Gather SCA dependencies on project
[INFO]  ScannerEngine: Dependency analysis skipped
[INFO]  ScannerEngine: SCM Publisher No SCM system was detected. You can use the 'sonar.scm.provider' property to explicitly specify it.
[INFO]  ScannerEngine: CPD Executor 2 files had no CPD blocks
[INFO]  ScannerEngine: CPD Executor Calculating CPD for 8 files
[INFO]  ScannerEngine: CPD Executor CPD calculation finished (done) | time=4ms
[INFO]  ScannerEngine: Analysis report generated in 39ms, dir size=298.6 kB
[INFO]  ScannerEngine: Analysis report compressed in 39ms, zip size=61.7 kB
[INFO]  ScannerEngine: Analysis report uploaded in 37ms
[INFO]  ScannerEngine: ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=FRONTEND
[INFO]  ScannerEngine: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
[INFO]  ScannerEngine: More about the report processing at http://localhost:9000/api/ce/task?id=35c8143c-1b8e-423b-a531-33245ef54736
[INFO]  ScannerEngine: Analysis total time: 4.735 s
[INFO]  ScannerEngine: SonarScanner Engine completed successfully
(base) samuels@MacBook-Pro-de-Samuel-2 frontend %
```




frontend
Public
Passed
Last analysis: 1 minute ago • 449 Lines of Code • TypeScript, CSS, ...
A 0
Security
C 6
Reliability
A 2
Maintainability
A —
Hotspots Reviewed
0.0%
Coverage
0.0%
Duplications
Maintainability
neocdt
Public
Passed
Last analysis: 5 minutes ago • 219 Lines of Code • JavaScript
A 0
Security
A 0
Reliability
A 3
Maintainability
E 0.0%
Hotspots Reviewed
0.0%
Coverage
0.0%
Duplications

Backend


localhost
Google
Apple
Yahoo
Gmail
YouTube
Maps
Noticias
Traducir
Embedded database should be used for evaluation purposes only. It doesn't support scaling, upgrading to a new SonarQube Server version, or migration to another database engine. Learn more
SonarQube community
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
More
neocdt
Bind project
main
Overview
Issues
Security Hotspots
Code
Measures
Activity
Project Settings
Project Information
Quality Gate
Passed
Last analysis 1 minute ago
The last analysis has warnings. See details
New Code
Overall Code
Security
0 Open issues
A
Reliability
0 Open issues
A
Maintainability
3 Open issues
A
Accepted issues
0
Valid issues that were not fixed
Coverage
0.0%
On 134 lines to cover.
Duplications
0.0%
On 292 lines.
Security Hotspot
1
E

```
[INFO]  ScannerEngine: Sensor IaC Azure Resource Manager Sensor [iac]
[INFO]  ScannerEngine: There are no files to be analyzed for the Azure Resource Manager language
[INFO]  ScannerEngine: Sensor IaC Azure Resource Manager Sensor [iac] (done) | time=0ms
[INFO]  ScannerEngine: Sensor Java Config Sensor [iac]
[INFO]  ScannerEngine: There are no files to be analyzed for the Java language
[INFO]  ScannerEngine: Sensor Java Config Sensor [iac] (done) | time=1ms
[INFO]  ScannerEngine: Sensor HTML [web]
[INFO]  ScannerEngine: Sensor HTML [web] (done) | time=1ms
[INFO]  ScannerEngine: Sensor JavaScript/TypeScript analysis [javascript]
[INFO]  ScannerEngine: Detected os: Mac OS X arch: aarch64 alpine: false. Platform: DARWIN_ARM64
[INFO]  ScannerEngine: Deploy location /Users/samuels/.sonar/js/node-runtime, tagetRuntime: /Users/samuels/.sonar/js/node-runtime/node,  version: /Users/samuels/.sonar/js/node-runtime/version.txt
[INFO]  ScannerEngine: Using embedded Node.js runtime.
[INFO]  ScannerEngine: Using Node.js executable: '/Users/samuels/.sonar/js/node-runtime/node'.
[INFO]  ScannerEngine: Memory configuration: OS (8192 MB), Node.js (2096 MB).
[INFO]  ScannerEngine: WebSocket client connected on /ws
[INFO]  ScannerEngine: Plugin version: [11.4.1.34873]
[INFO]  ScannerEngine: Using generated tsconfig.json file using wildcards /private/var/folders/d2/m227k6t53cb_vm5xzcyp4mf0000gn/T/tsconfig-feXxh1.json
[INFO]  ScannerEngine: Found 1 tsconfig.json file(s): [/private/var/folders/d2/m227k6t53cb_vm5xzcyp4mf0000gn/T/tsconfig-feXxh1.json]
[INFO]  ScannerEngine: 11 source files to be analyzed
[INFO]  ScannerEngine: Creating TypeScript program
[INFO]  ScannerEngine: TypeScript(5.9.2) configuration file /private/var/folders/d2/m227k6t53cb_vm5xzcyp4mf0000gn/T/tsconfig-feXxh1.json
[INFO]  ScannerEngine: Found 3 file(s) not part of any tsconfig.json: they will be analyzed without type information
[INFO]  ScannerEngine: 11/11 source files have been analyzed
[INFO]  ScannerEngine: Hit the cache for 0 out of 11
[INFO]  ScannerEngine: Miss the cache for 11 out of 11: ANALYSIS_MODE_INELIGIBLE [11/11]
[INFO]  ScannerEngine: Sensor JavaScript/TypeScript analysis [javascript] (done) | time=7111ms
[INFO]  ScannerEngine: Sensor CSS Rules [javascript]
[INFO]  ScannerEngine: No CSS, PHP, HTML or VueJS files are found in the project. CSS analysis is skipped.
[INFO]  ScannerEngine: Sensor CSS Rules [javascript] (done) | time=0ms
[INFO]  ScannerEngine: Sensor TextAndSecretsSensor [text]
[INFO]  ScannerEngine: Available processors: 8
[INFO]  ScannerEngine: Using 8 threads for analysis.
[INFO]  ScannerEngine: The property "sonar.tests" is not set. To improve the analysis accuracy, we categorize a file as a test file if any of the following is true:
  * The filename starts with "test"
  * The filename contains "test." or "tests."
  * Any directory in the file path is named: "doc", "docs", "test" or "tests"
  * Any directory in the file path has a name ending in "test" or "tests"

[INFO]  ScannerEngine: Start fetching files for the text and secrets analysis
[INFO]  ScannerEngine: Using Git CLI to retrieve untracked files
[WARN]  ScannerEngine: Retrieving only language associated files, make sure to run the analysis inside a git repository to make use of inclusions specified via "sonar.text.inclusions"
[INFO]  ScannerEngine: Starting the text and secrets analysis
[INFO]  ScannerEngine: 12 source files to be analyzed for the text and secrets analysis
[INFO]  ScannerEngine: 12/12 source files have been analyzed for the text and secrets analysis
[INFO]  ScannerEngine: Sensor TextAndSecretsSensor [text] (done) | time=332ms
[INFO]  ScannerEngine: ------------- Run sensors on project
[INFO]  ScannerEngine: Sensor Zero Coverage Sensor
[INFO]  ScannerEngine: Sensor Zero Coverage Sensor (done) | time=4ms
[INFO]  ScannerEngine: ------------- Gather SCA dependencies on project
[INFO]  ScannerEngine: Dependency analysis skipped
[INFO]  ScannerEngine: SCM Publisher No SCM system was detected. You can use the 'sonar.scm.provider' property to explicitly specify it.
[INFO]  ScannerEngine: CPD Executor 3 files had no CPD blocks
[INFO]  ScannerEngine: CPD Executor Calculating CPD for 8 files
[INFO]  ScannerEngine: CPD Executor CPD calculation finished (done) | time=7ms
[INFO]  ScannerEngine: Analysis report generated in 44ms, dir size=289.5 kB
[INFO]  ScannerEngine: Analysis report compressed in 43ms, zip size=57.6 kB
[INFO]  ScannerEngine: Analysis report uploaded in 143ms
[INFO]  ScannerEngine: ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=NEOCDT
[INFO]  ScannerEngine: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
[INFO]  ScannerEngine: More about the report processing at http://localhost:9000/api/ce/task?id=7c8ed4f2-94b9-4bce-8929-d225f1dc6640
[INFO]  ScannerEngine: Analysis total time: 10.035 s
[INFO]  ScannerEngine: SonarScanner Engine completed successfully
(base) samuels@MacBook-Pro-de-Samuel-2 backend % 
```

```
[INFO]  Bootstrapper: SonarQube server version: 25.10.0
[INFO]  Bootstrapper: JRE provisioning is supported
[INFO]  Bootstrapper: No Cache found for JRE
[INFO]  Bootstrapper: Download starting...
[INFO]  Bootstrapper: Download complete
[INFO]  Bootstrapper: Downloaded JRE to /Users/samuels/.sonar/cache/a886b8f2a50eca2e59b45ea59f5a2e8e9d27ff5b5b3b069443a70cda7f27c907/OpenJDK17U-jre_aarch64_mac_hotspot_17.0.13_11.tar.gz
[INFO]  Bootstrapper: No Cache found for SonarScanner Engine
[INFO]  Bootstrapper: Download starting...
[INFO]  Bootstrapper: Download complete
[INFO]  Bootstrapper: Downloaded SonarScanner Engine to /Users/samuels/.sonar/cache/5370664af11ed9502c5e9da2fca3e7b2465b6b01c7248962ef899d0551837b42/sonar-scanner-engine-shaded-25.10.0.114319-all.jar
[INFO]  ScannerEngine: Starting SonarScanner Engine...
[INFO]  ScannerEngine: Java 17.0.13 Eclipse Adoptium (64-bit)
[INFO]  ScannerEngine: Load global settings
[INFO]  ScannerEngine: Load global settings (done) | time=102ms
[INFO]  ScannerEngine: Server id: 147B411E-AZnvUbbk7ub8yistJEm2
[INFO]  ScannerEngine: Loading required plugins
[INFO]  ScannerEngine: Load plugins index
[INFO]  ScannerEngine: Load plugins index (done) | time=34ms
[INFO]  ScannerEngine: Load/download plugins
[INFO]  ScannerEngine: Load/download plugins (done) | time=306ms
[INFO]  ScannerEngine: Process project properties
[INFO]  ScannerEngine: Process project properties (done) | time=0ms
[INFO]  ScannerEngine: Project key: NEOCDT
[INFO]  ScannerEngine: Base dir: /Users/samuels/Desktop/NEOCDT-main/neocdt/backend
[INFO]  ScannerEngine: Working dir: /Users/samuels/Desktop/NEOCDT-main/neocdt/backend/.scannerwork
[INFO]  ScannerEngine: Load project settings for component key: 'NEOCDT'
[INFO]  ScannerEngine: Load project settings for component key: 'NEOCDT' (done) | time=20ms
[INFO]  ScannerEngine: Load quality profiles
[INFO]  ScannerEngine: Load quality profiles (done) | time=122ms
[WARN]  ScannerEngine: SCM provider autodetection failed. Please use "sonar.scm.provider" to define SCM of your project, or disable the SCM Sensor in the project settings.
[INFO]  ScannerEngine: Load active rules
[INFO]  ScannerEngine: Load active rules (done) | time=295ms
[INFO]  ScannerEngine: Load analysis cache
[INFO]  ScannerEngine: Load analysis cache (404) | time=8ms
[INFO]  ScannerEngine: Preprocessing files...
[INFO]  ScannerEngine: 2 languages detected in 14 preprocessed files (done) | time=13ms
[INFO]  ScannerEngine: 0 files ignored because of inclusion/exclusion patterns
[INFO]  ScannerEngine: Loading plugins for detected languages
[INFO]  ScannerEngine: Load/download plugins
[INFO]  ScannerEngine: Load/download plugins (done) | time=1111ms
[INFO]  ScannerEngine: Load project repositories
[INFO]  ScannerEngine: Load project repositories (done) | time=78ms
[INFO]  ScannerEngine: Indexing files...
[INFO]  ScannerEngine: Project configuration:
[INFO]  ScannerEngine:   Excluded sources: node_modules/**, bower_components/**, jspm_packages/**, typings/**, lib-cov/**
[INFO]  ScannerEngine: 14 files indexed (done) | time=6ms
[INFO]  ScannerEngine: Quality profile for js: Sonar way
[INFO]  ScannerEngine: Quality profile for json: Sonar way
[INFO]  ScannerEngine: ------------- Run sensors on module neocdt
[INFO]  ScannerEngine: Load metrics repository
[INFO]  ScannerEngine: Load metrics repository (done) | time=21ms
[INFO]  ScannerEngine: Sensor JaCoCo XML Report Importer [jacoco]
[INFO]  ScannerEngine: 'sonar.coverage.jacoco.xmlReportPaths' is not defined. Using default locations: target/site/jacoco/jacoco.xml,target/site/jacoco-it/jacoco.xml,build/reports/jacoco/test/jacocoTestRe
port.xml
[INFO]  ScannerEngine: No report imported, no coverage information will be imported by JaCoCo XML Report Importer
[INFO]  ScannerEngine: Sensor JaCoCo XML Report Importer [jacoco] (done) | time=1ms
[INFO]  ScannerEngine: Sensor IaC CloudFormation Sensor [iac]
[INFO]  ScannerEngine: There are no files to be analyzed for the CloudFormation language
[INFO]  ScannerEngine: Sensor IaC CloudFormation Sensor [iac] (done) | time=6ms
[INFO]  ScannerEngine: Sensor IaC cfn-lint report Sensor [iac]
[INFO]  ScannerEngine: Sensor IaC cfn-lint report Sensor [iac] (done) | time=0ms
[INFO]  ScannerEngine: Sensor IaC Docker Sensor [iac]
[INFO]  ScannerEngine: There are no files to be analyzed for the Docker language
```