ANALISTA
PROGRAMADOR

inacap

Unidad 2: Diseño de la solución de software.

NOMBRE: Claudio Quilodrán – Gonzalo Rodríguez – Javiera Sepúlveda.
CARRERA: Analista Programador.
ASIGNATURA: Proyecto Integrado.
PROFESOR: Sebastián Callejas.
FECHA: 27 de octubre de 2025.

1 Índice.

Contenido

2 Introducción.

El presenta informe corresponde a la segunda unidad de Proyecto Integrado, centrado principalmente en el diseño técnico del sistema NUAM, un mantenedor de calificaciones tributarias desarrollado bajo las metodologías Scrum y Design Thinking. El objetivo es definir la arquitectura, base de datos, interfaz, políticas de seguridad y plan de pruebas del sistema, cumpliendo con las normativas chilenas vigentes: Ley 19.628 sobre protección de datos personales, Ley 21.663 y el Decreto 7/2023 del Ministerio de Hacienda.

El proyecto adopta la combinación de las metodologías ya mencionadas para permitir un enfoque iterativo y centrado en el usuario. Este modelo favorece la validación temprana de prototipos, la corrección continua y la priorización de requerimientos según valor de negocio y complejidad técnica.

3 Metodología y cronograma de trabajo.

3.1 Justificación de la metodología

En primer lugar, se eligieron la combinación de metodologías **Scrum + Design Thinking**, gestionando la planificación en la plataforma **Taiga**, que da una buena integración entre backlog, sprint y Kanban. Además de ser fácil de usar para equipos pequeños y soporte de épicas.

- ***Scrum*** permite entregas incrementales y visibilidad frecuente de valor (sprints de 2 semanas), ideal para proyectos con requisitos que pueden refinarse tras feedback. NUAM necesita validar procesos regulados (auditoría, carga masiva) con usuarios (analistas, auditores), por lo que iterar reduce riesgo de reprocesos.
- ***Design Thinking*** aporta técnicas centradas en usuarios (entrevistas, prototipos), que son imprescindibles para interfaces de carga masiva y usabilidad en auditoría. Este método se compone de cinco fases que no siempre son lineales, donde el objetivo es comprender al usuario y diseñar soluciones útiles, viables y sostenibles:
    1. Empatizar: entrevistas con analistas y auditores tributarios para entender sus problemas como la carga de datos manual lenta, errores humanos, dificultad para auditar.
    2. Definir: sintetizar la información y formular un problema -> “Se necesita un sistema que reduzca errores y tiempos de carga de calificaciones tributarias, asegurando trazabilidad y cumplimiento legal.”
    3. Idear: generar posibles soluciones con *Brainstorming* en plataformas como Taiga o Miro, luego priorizar las más viables. Ideas como carga masiva, reportes automáticos, validación en tiempo real son las que se proponen en el proyecto.
    4. Prototipar: crear versiones tempranas de la solución para probar conceptos con *Mockups* de la interfaz usando wireframes.
    5. Evaluar/Testear: probar el prototipo con usuarios reales, obtener feedback y mejorar.

La complementación de estas dos metodologías nos da como resultado el cómo se organiza el trabajo por parte de Scrum, y el por qué vale la pena construirlo por parte del Design Thinking, donde cada fase se puede repetir dentro de cada sprint. Generando un desarrollo ágil, iterativo y centrado en el valor real para el usuario.

3.2 Product Backlog y plan de sprints

Se definen las historias de usuario con épicas y criterios de aceptación, además de implementar y programar los sprints.

Historias de Usuario, Proyecto: NUAM

ID	Título	Épica	Rol	Criterios de aceptación	Descripción	Prioridad	Estado
HU-01	**Carga masiva**	Centralización de datos	Analista	1. El sistema debe permitir cargar un archivo CSV o Excel y validar su formato antes de procesarlo 2. Al finalizar la carga, el sistema debe mostrar un resumen con el total de registros procesados, aceptados y rechazados	Como analista quiero cargar archivos masivos para automatizar procesos.	Must have	Pendiente
HU-04	**Ingreso manual**	Centralización de datos	Analista	1. El formulario debe validar campos obligatorios antes de guardar 2. El registro debe reflejarse inmediatamente en la base de datos y mostrarse en el listado general	Como analista quiero ingresar calificaciones manualmente para actualizar datos.	Must have	Pendiente
HU-05	**Eliminación/modificación**	Centralización de datos	Analista	1. Solo los usuarios con rol *analista* o superior pueden editar o eliminar registros 2. Toda modificación o eliminación debe quedar registrada en alguna tabla de auditoría	Como analista quiero modificar/eliminar calificaciones para corregir errores.	Must have	Pendiente
HU-03	**Gestión de accesos**	Seguridad	Administrador	1. El administrador debe poder asignar, modificar o revocar roles desde el panel de administración 2. Los cambios en los roles deben registrarse automáticamente en los logs de auditoría	Como administrador quiero gestionar permisos para controlar roles y auditoría.	Must have	Pendiente
HU-02	**Consulta histórica**	Auditoría y trazabilidad	Auditor	1. El sistema debe permitir filtrar auditorías por usuario, fecha o tipo de acción 2. Los resultados deben presentarse en orden cronológico y exportarse a CSV	Como auditor quiero consultar registros históricos para verificar cumplimiento.	Should have	Pendiente
HU-06	**Reportes auditoría**	Auditoría y trazabilidad	Auditor	1. El sistema debe generar un reporte consolidado con todas las modificaciones de registros 2. El reporte debe incluir usuario responsible, fecha y tipo de cambio	Como auditor quiero generar reportes de cambios para asegurar trazabilidad.	Should have	Pendiente
HU-08	**Exportación de datos**	Auditoría y trazabilidad	Analista	1. El sistema debe permitir exportar los datos visibles en pantalla en formato CSV o Excel 2. El archivo exportado debe incluir los mismos filtros y orden aplicados en la vista	Como analista quiero exportar calificaciones en CSV/Excel para otros sistemas.	Should have	Pendiente
HU-07	**Notificaciones de cambios**	Seguridad y alertas	Administrador	1. El sistema debe enviar un correo o notificación interna cuando se modifiquen o eliminen los registros sensibles 2. Las notificaciones deben agruparse por usuario y tipo de evento	Como administrador quiero recibir alertas de cambios en calificaciones.	Could have	Pendiente
HU-09	**Dashboard de métricas**	Monitoreo y control	Gerente	1. El dashboard debe mostrar al menos tres métricas clave: cantidad de operaciones, errores y tiempos de carga 2. Las métricas deben actualizarse en tiempo real o cada 10 minutos	Como gerente quiero ver métricas de uso y errores para evaluar desempeño.	Could have	Pendiente

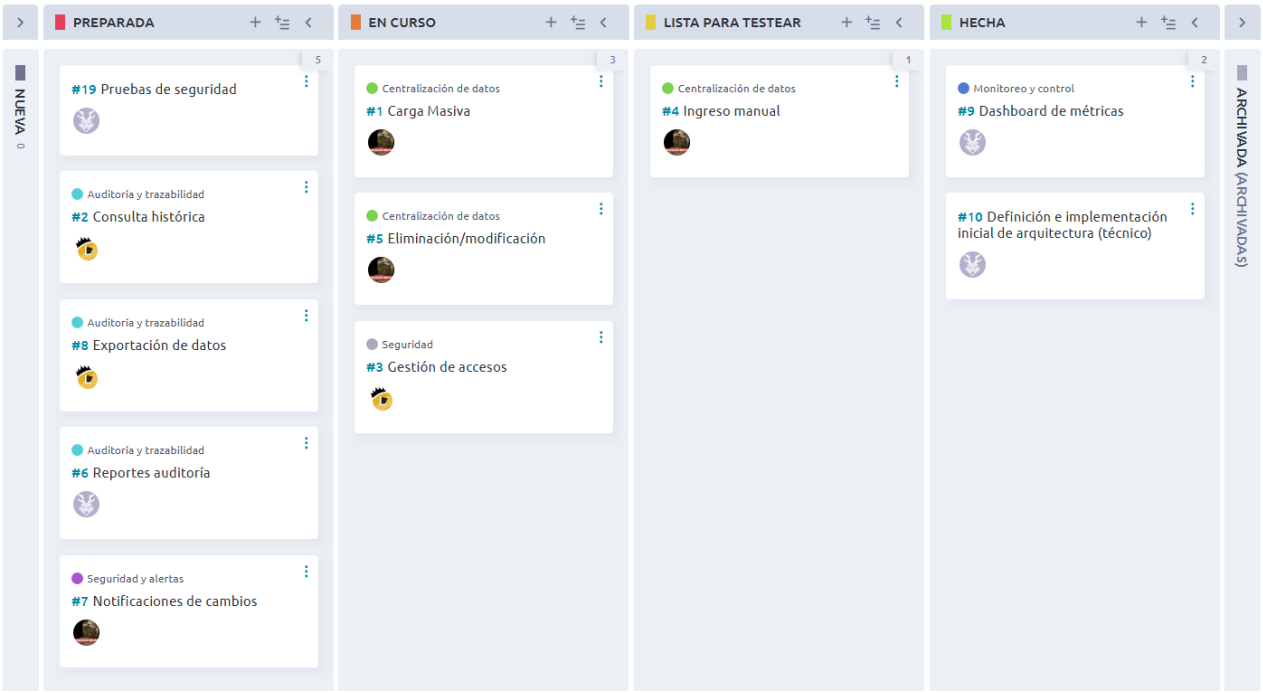
Sprint	Objetivo principal	Entregables	Responsables
Sprint 1	Análisis técnico y arquitectura base	Documento de arquitectura, diagramas	Equipo completo
Sprint 2	CRUD y roles	Módulo de usuarios y calificaciones	Desarrolladores
Sprint 3	Carga masiva y reportes	Carga masiva y generación de informes	Backend y QA
Sprint 4	Pruebas, seguridad y despliegue	Plan de pruebas, hardening y despliegue final	Todos los miembros

Durante el desarrollo, se realizaron ceremonias Scrum: *daily meeting* para seguimiento, *sprint review* para validar avances y *retrospective* para mejorar procesos.

Cada sprint tuvo una duración de dos semanas (10 días hábiles) y contó con los siguientes roles:

- Product Owner: encargado de priorizar el backlog y validar entregables.
- Scrum Master: responsable de facilitar reuniones y remover impedimentos.
- Development Team: equipo encargado del diseño, desarrollo y pruebas.

3.3 Tablero Kanban



Proceso actual del trabajo en tablero Kanban para mantener el flujo con cuatro estados: *Preparada, En curso, Lista para testear y Hecha*. Cada historia de usuario tiene definida tareas.

Sprint 1:

- Ingreso manual: CRUD base.
- Dashboard de métricas: mockups de interfaz (dashboard, login, gestión)
- Definición e implementación inicial de arquitectura: diagramas del paradigma 4+1, diseño y estructura de la base de datos.

Sprint 2:

- Eliminación / Modificación: implementación de actualización y eliminación de calificaciones (CRUD avanzado)
- Gestión de accesos: integración del sistema de roles (administrador, analista, auditor), validaciones de acceso y logging de acciones.

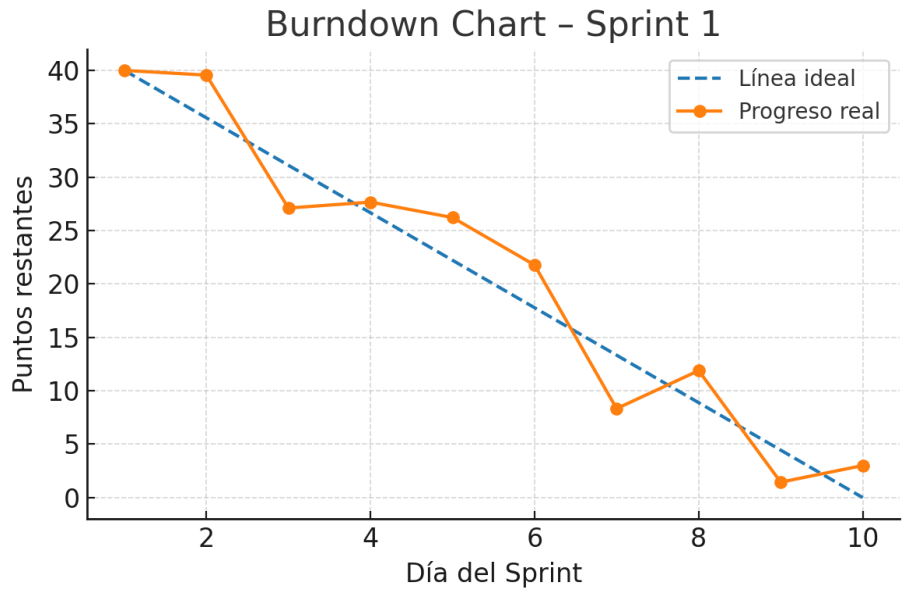
Sprint 3:

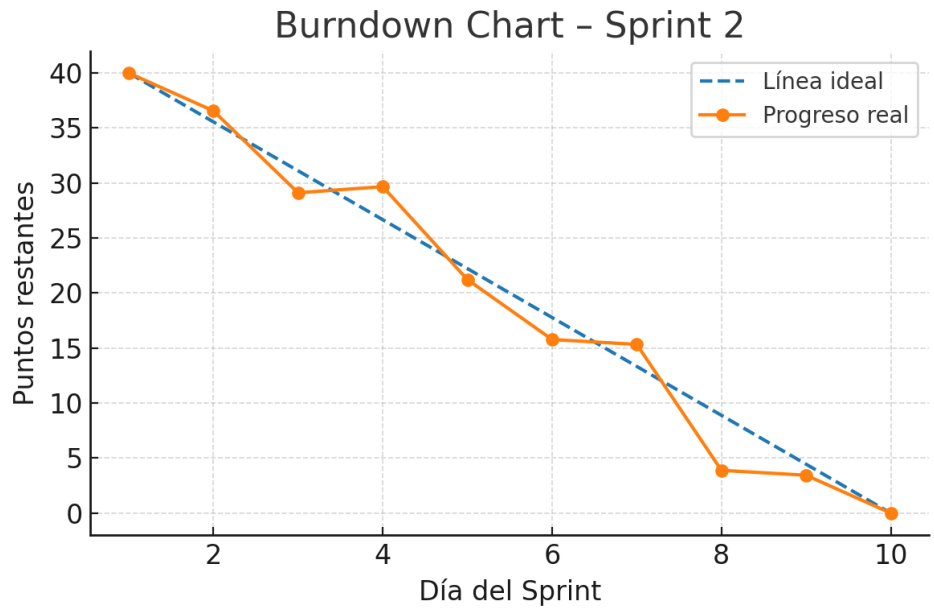
- Carga masiva: módulo de carga masiva con validación de archivos CSV/XLSX.
- Reportes de auditoría: generación de reportes.
- Exportación de datos: permitir exportación de calificaciones a Excel/CSV.

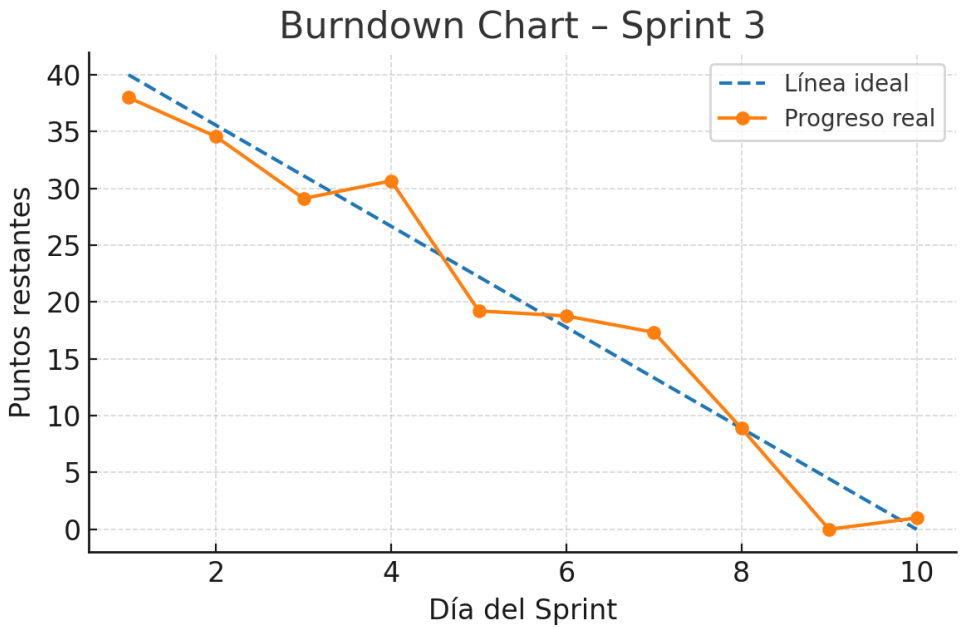
Sprint 4:

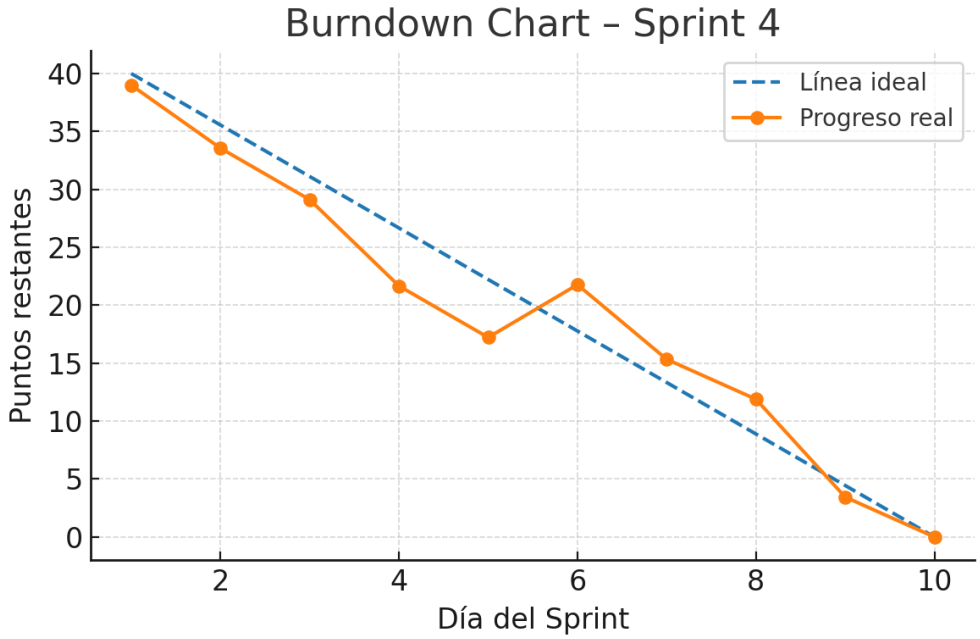
- Pruebas de seguridad: implementación del plan de pruebas.
- Notificaciones de cambio: módulo de notificaciones ante cambios en el sistema.
- Consulta histórica: vista de consulta histórica filtrada y exportable.

A continuación, se muestran los gráficos *Burndown* que representan el trabajo restante de cada Sprint.






Burndown Chart – Sprint 3
Línea ideal
Progreso real
Puntos restantes
Día del Sprint
0
5
10
15
20
25
30
35
40
2
4
6
8
10

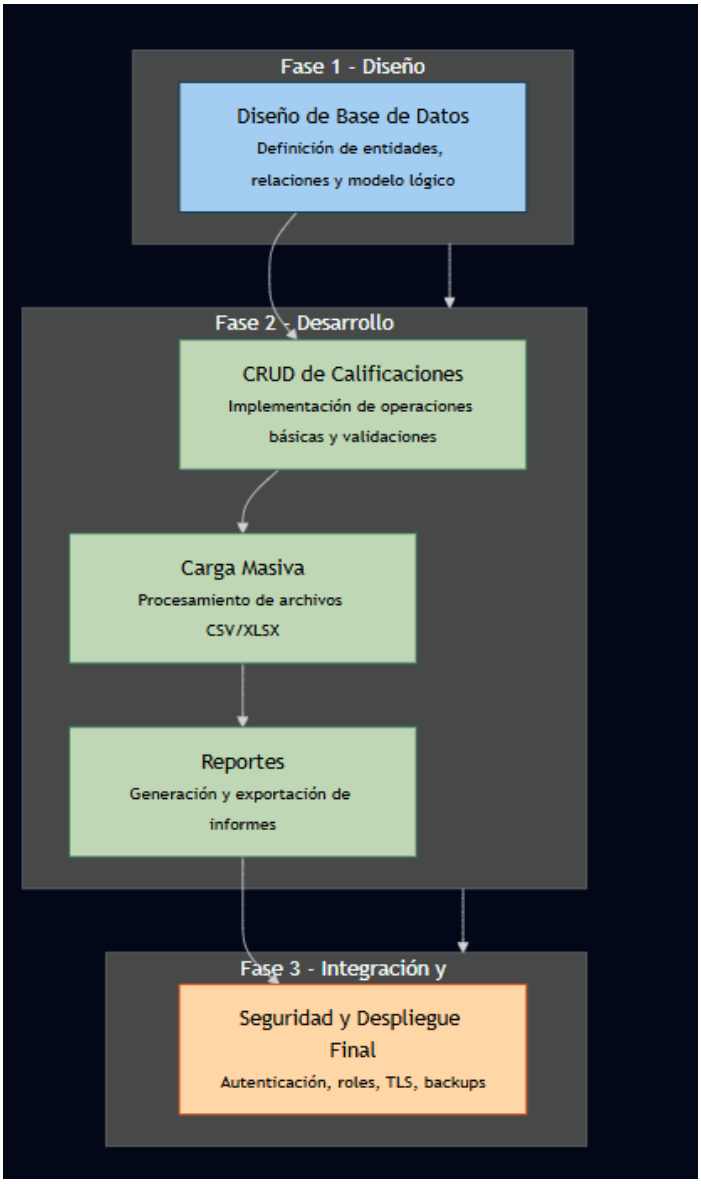

Burndown Chart – Sprint 4
Línea ideal
Progreso real
Puntos restantes
Día del Sprint
0
5
10
15
20
25
30
35
40
2
4
6
8
10

3.4 Dependencias y prioridades

Tarea	Depende de	Prioridad
Diseño BD	Ninguna	Alta
CRUD	Diseño BD	Alta
Carga masiva	CRUD	Media
Reportes	Carga masiva	Media
Seguridad y despliegue	Todas las anteriores	Alta

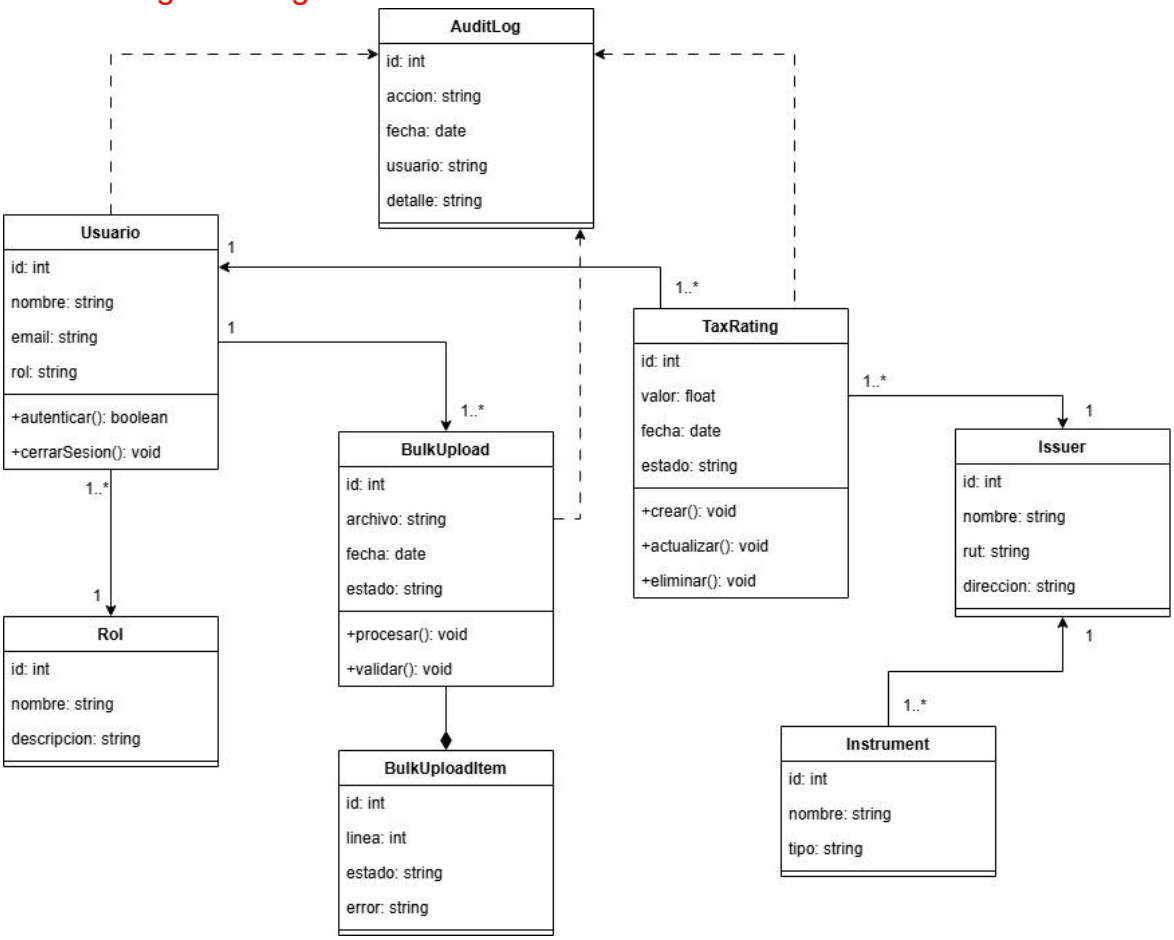
Las dependencias se gestionan mediante herramientas colaborativas como **Jira** o **Trello**, donde cada tarea del Product Backlog se vincula a sus predecesoras.

El equipo prioriza las tareas según impacto y complejidad, aplicando el método **MoSCoW (Must, Should, Could, Won't)** para optimizar la entrega incremental.

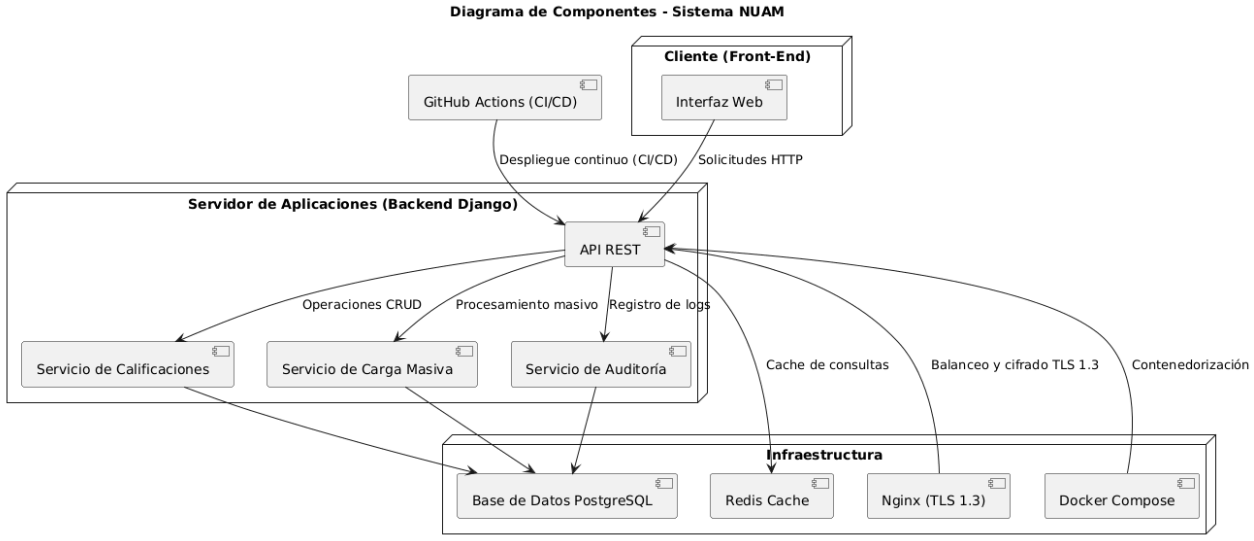


4 Paradigma 4+1

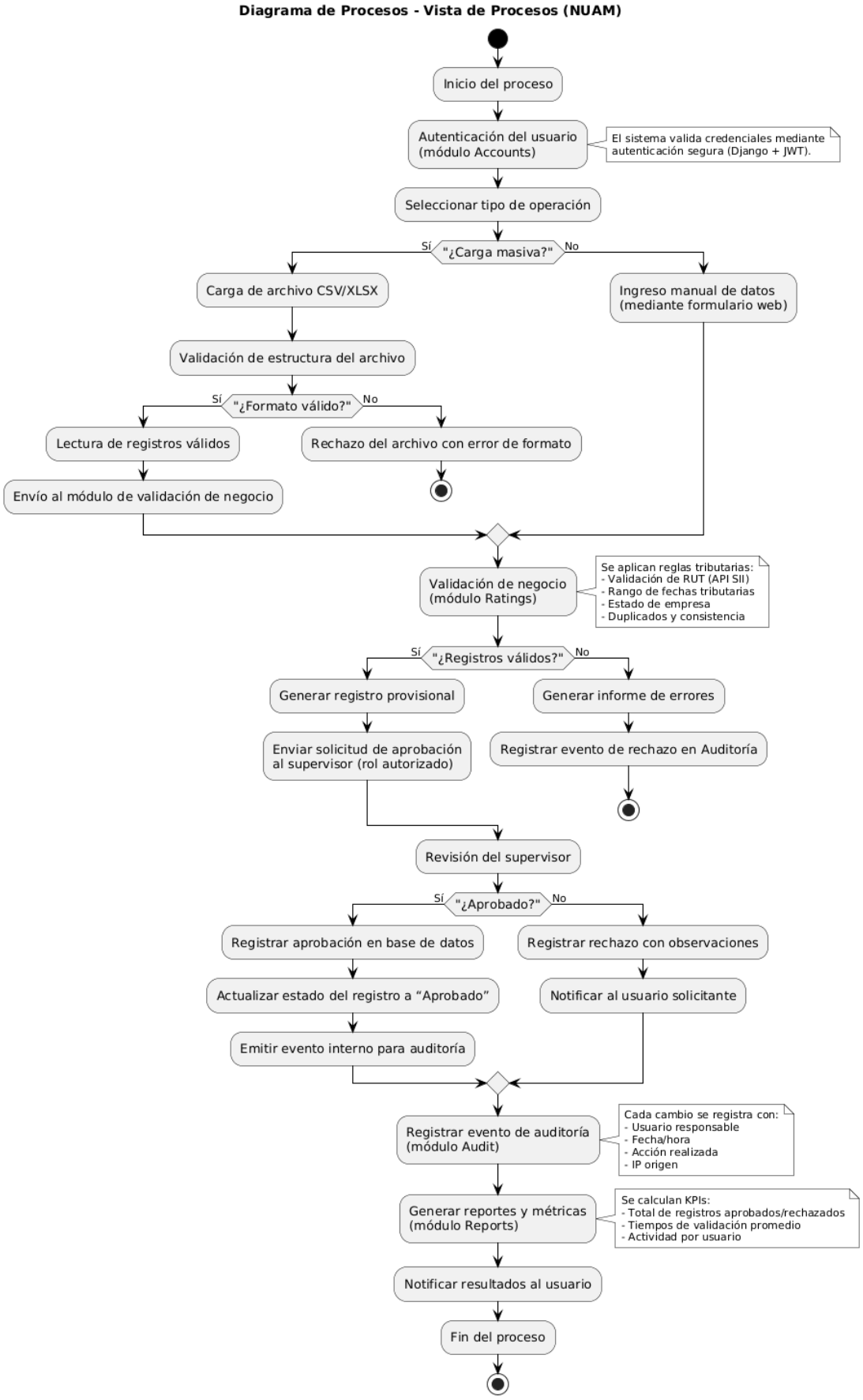
4.1 Vista Lógica: Diagrama de clases



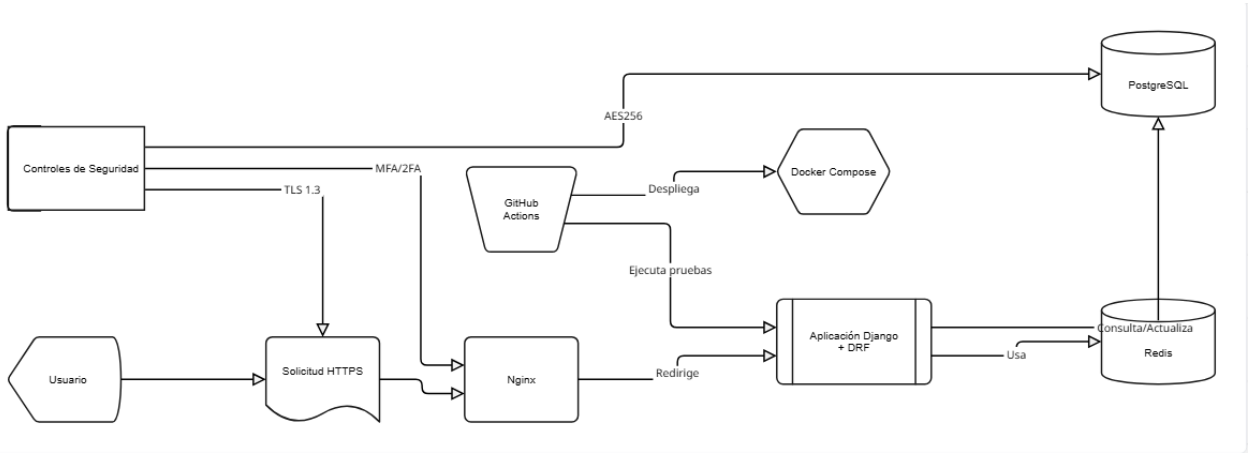
4.2 Vista de Desarrollo: Diagrama de componentes



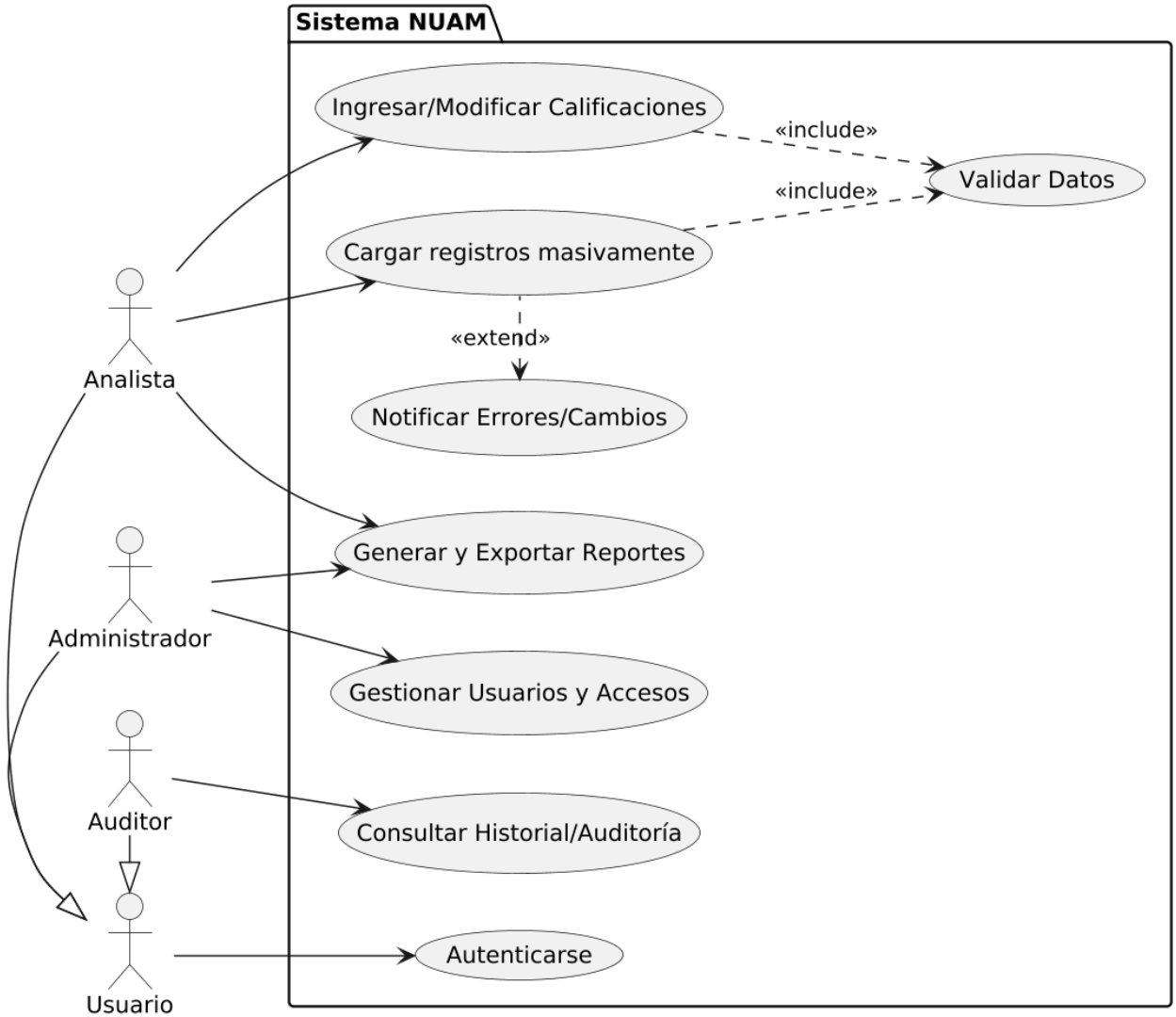
4.3 Vista de Procesos: BPMN



4.4 Vista Física: Diagrama de topología



4.5 Vista de Escenarios: Casos de uso

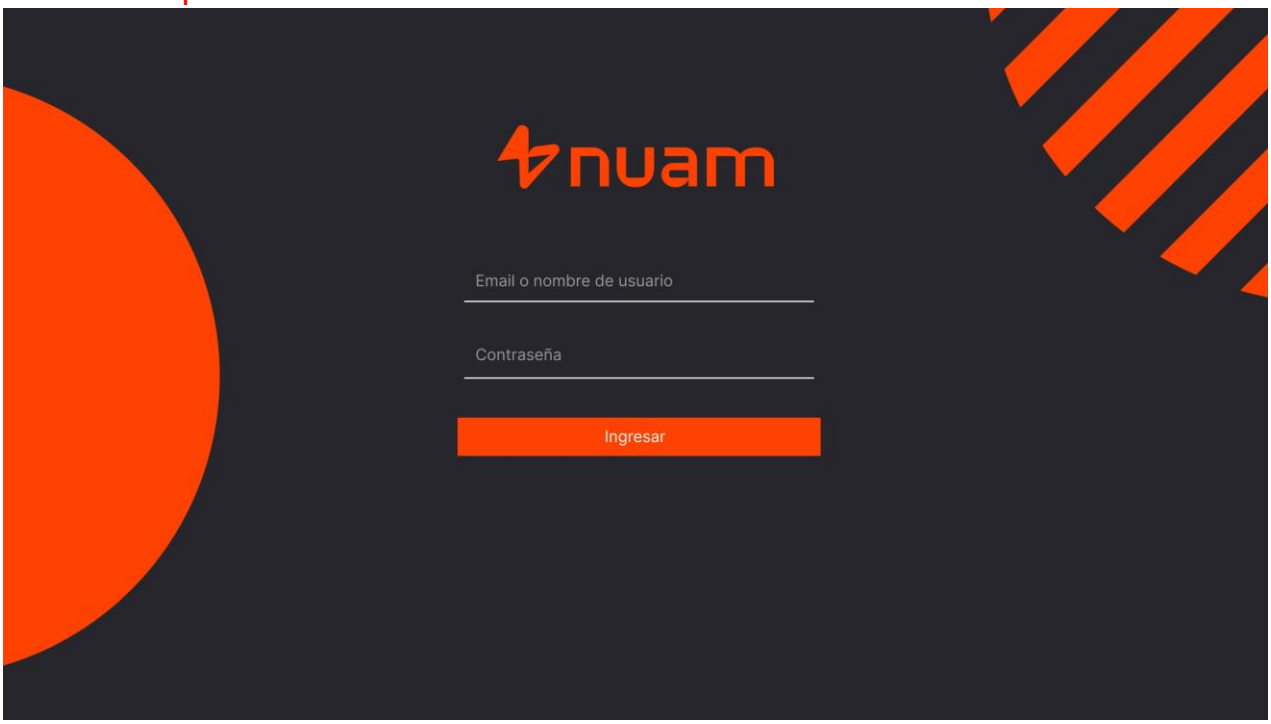


5 Diseño de interfaz y experiencia de usuario (UI/UX)

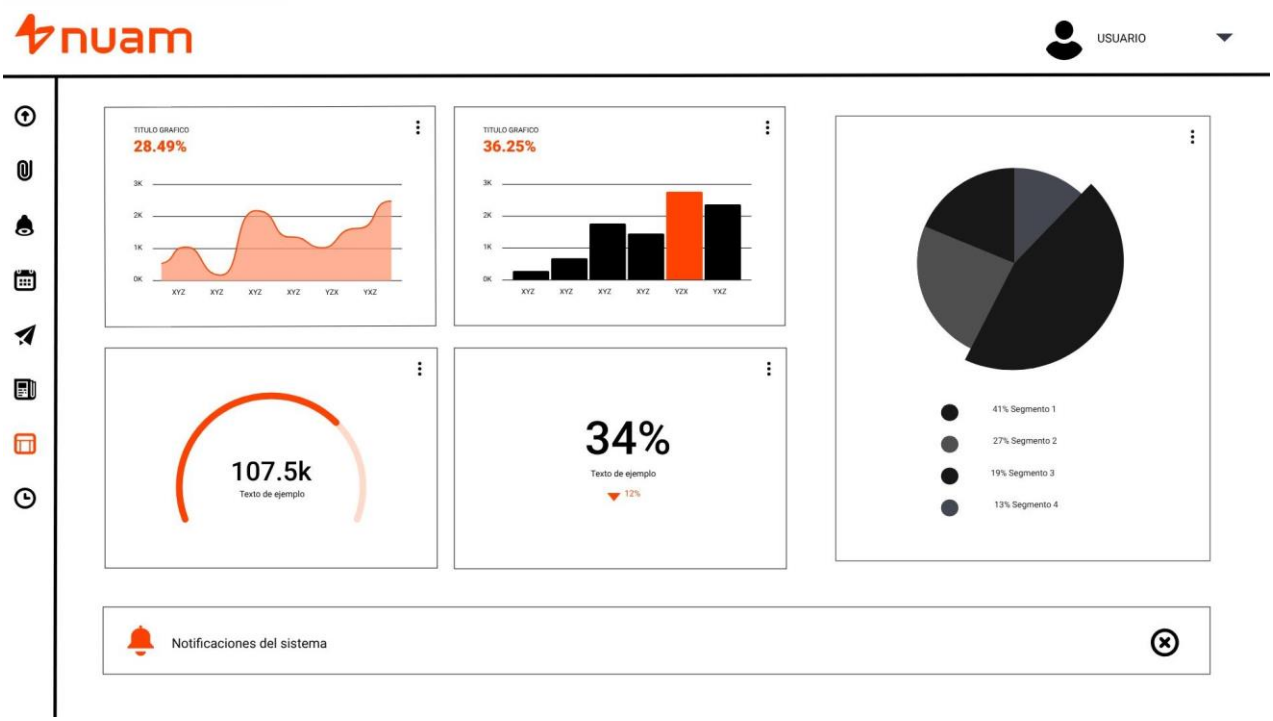
Pantalla	Objetivo	Elementos clave
Login	Permitir acceso seguro	Campos usuario/contraseña, botón ingresar
Dashboard	Vista general	Panel con métricas y accesos rápidos
Gestión de calificaciones	CRUD de calificaciones tributarias	Tablas editables, filtros y botones de acción
Carga masiva	Registrar múltiples registros	Selector de archivo, barra de progreso, validaciones
Reportes	Visualizar informes	Gráficos, filtros por fecha y exportación PDF

El diseño del flujo de navegación se estructura en un mapa visual de sitio jerárquico, donde el dashboard es el punto de acceso principal. Además, se cumple con el estándar de accesibilidad **WCAG AA**. Los mockups se realizaron como prototipos, priorizando la funcionalidad y el diseño visual intuitivo para el usuario.

5.1 Mockups



Login



Dashboard de métricas

Módulos de gestión, reportes y carga masiva

6 Modelo de datos y almacenamiento

Se utiliza el ORM de Django con una base de datos PostgreSQL, normalizada en tercera forma normal (3FN), manejo de claves foráneas, auditorías y soporte JSON para estructuras dinámicas.

Entidad	Descripción
Issuer	Entidad emisora o contribuyente.
Instrument	Instrumentos financieros asociados.
TaxRating	Registro de calificación tributaria.
BulkUpload	Registro de carga masiva.
BulkUploadItem	Detalle de elementos cargados en bloque.
AuditLog	Registro de auditoría en formato JSON.
User	Usuarios del sistema con roles asignados.

Características técnicas:

- Integridad referencial mediante *foreign keys* con reglas **ON DELETE/UPDATE CASCADE.**
- Auditorías automáticas en formato JSON, lo que facilita trazabilidad y análisis histórico.
- Diseño híbrido que permite incorporar datos estructurados (relacionales) y no estructurados (JSONB)

7 Estándares de programación segura

Se implementan políticas de codificación segura siguiendo las buenas prácticas **PEP8 + OWASP Top 10**, priorizando integridad, confidencialidad y disponibilidad de los datos. Cumpliendo así las normativas chilenas de protección de datos, ciberseguridad y el decreto 7/2023.

- PEP8 proporciona reglas y convenciones de formato para que el código Python sea legible y consistente. El objetivo es la claridad y colaboración entre desarrolladores, ya que se sigue un conjunto de estándares de estilos comunes como *snake_case* para nombres de variables o *UpperCamelCase* para nombres de clases, entre otras cosas.
- OWASP Top 10 es un documento de concientización sobre los diez riesgos de seguridad más críticos para las aplicaciones web. Funciona como guía estándar para desarrolladores, ayudando a identificar y mitigar las amenazas.

7.1 Roles y privilegios (RBAC)

Se identifican los perfiles de usuarios que interactúan con el sistema basado en el *control de acceso basado en roles.*

Rol	Descripción	Permisos principales
Administrador	Supervisa el sistema completo	Crear, editar y eliminar usuarios, revisar logs
Analista	Gestiona calificaciones	CRUD de calificaciones y generación de reportes
Auditor	Supervisa integridad de datos	Acceso de solo lectura a registros y reportes

7.2 Amenazas, medidas de mitigación y estándares de codificación segura

Se identifican distintos tipos de amenazas y propuestas para mitigarlas, reduciendo vulnerabilidades y cumpliendo los requisitos del decreto 7/2023 del Ministerio de Hacienda.

Amenaza	Medida de mitigación
SQL Injection	Uso de ORM, validación y sanitización de datos.
XSS/CSRF	Middleware de protección y validación de tokens.
Fuga de datos	Cifrado AES-256 y control de permisos.
Accesos no autorizados	Roles RBAC y autenticación MFA.
Pérdida de información	Backups automáticos y logs inmutables.

Autenticación MFA + expiración de sesión.

Autorización por roles (RBAC).

Validación de entradas y sanitización de datos.

Cifrado en tránsito (TLS) y en reposo (PostgreSQL + AES-256).

Logs inmutables y auditoría.

Backups automáticos y restauración validada.

8 Plan de pruebas

Se propone un plan de pruebas que garantiza la estabilidad y seguridad del sistema, siguiendo las recomendaciones de OWASP y las buenas prácticas de QA. El flujo CI/CD en GitHub Actions automatiza la ejecución de pruebas y despliegues, alcanzando una cobertura que cumple los estándares de aseguramiento de calidad definidos para proyectos Django.

Tipo de prueba	Objetivo	Herramienta
Funcional	Verificar funciones individuales	pytest-django
Integración	Validar comunicación entre módulos	Django Test Client
Seguridad	Detectar vulnerabilidades OWASP	OWASP ZAP
Rendimiento	Evaluar carga y tiempos de respuesta	locust.io
Usabilidad	Evaluar experiencia del usuario	Selenium

8.1 Casos de prueba

ID	Tipo	Objetivo	Datos / Entrada	Pasos	Resultado esperado	Herramienta
CP-01	Funcional / Seguridad	Verificar autenticación y roles.	Credenciales válidas/erróneas.	Ingresar datos, acceder módulo restringido.	Acceso sólo a usuarios autorizados.	pytest-django, Selenium
CP-02	Funcional	Validar CRUD de calificaciones.	Datos completos/incompletos.	Crear, editar, eliminar registros.	CRUD exitoso y auditado.	pytest-django
CP-03	Integración / Rendimiento	Probar carga masiva (10k filas).	Archivo CSV mixto.	Subir y procesar.	Tiempo <5min, sin fallos críticos.	Locust
CP-04	Seguridad	Detectar vulnerabilidades OWASP.	Peticiones manipuladas.	Escanear con ZAP y probar inyecciones.	Sin vulnerabilidades críticas.	OWASP ZAP
CP-05	Usabilidad	Medir facilidad de uso.	Tarea práctica de carga y corrección.	Completar flujo completo.	Usuario termina en <6min, SUS ≥70.	Selenium / encuesta SUS

El plan de pruebas del sistema NUAM busca garantizar la funcionalidad, seguridad y rendimiento del software antes de su implementación definitiva. Se diseñaron pruebas orientadas a validar los módulos críticos (autenticación, CRUD, carga masiva y reportes) aplicando un enfoque mixto de validación manual y automatizada. Las herramientas utilizadas incluyen pytest-django, Selenium, Locust y OWASP ZAP, cubriendo pruebas funcionales, de integración, de seguridad, de rendimiento y de usabilidad. A continuación, se presentan los casos de prueba principales definidos para esta etapa.

9 Conclusiones

El diseño del sistema NUAM cumple con los estándares técnicos y legales establecidos, asegurando escalabilidad, trazabilidad y cumplimiento normativo. La metodología ágil Scrum, junto con Django y las herramientas de CI/CD, garantizan un desarrollo controlado, auditable y eficiente.

Este diseño refleja una planificación ordenada, cumpliendo con las normativas chilenas y los lineamientos técnicos de seguridad y escalabilidad.

El trabajo colaborativo bajo Scrum permitió integrar el enfoque ágil con la validación iterativa de usuarios, fortaleciendo la trazabilidad y la calidad del producto final.

10 Referencias

Metodologías ágiles y diseño centrado en el usuario

- Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., & Thomas, D. (2001). *Manifesto for Agile Software Development.* Agile Alliance. https://agilemanifesto.org/

- Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide: The Definitive Guide to Scrum.* Scrum.org. https://scrumguides.org/

- Brown, T. (2009). *Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation.* HarperCollins.

Arquitectura de software – Modelo 4+1

- Kruchten, P. (1995). *Architectural Blueprints—The "4+1" View Model of Software Architecture. IEEE Software*, 12(6), 42–50. https://doi.org/10.1109/52.469759

- Bass, L., Clements, P., & Kazman, R. (2021). *Software Architecture in Practice* (4th ed.). Addison-Wesley.

Infraestructura y tecnologías utilizadas

- Django Software Foundation. (2024). *Django Documentation (v5.0).* https://docs.djangoproject.com/en/5.0/

- PostgreSQL Global Development Group. (2023). *PostgreSQL 15 Documentation.* https://www.postgresql.org/docs/15/

- Docker, Inc. (2024). *Docker Documentation.* https://docs.docker.com/

- GitHub, Inc. (2024). *GitHub Actions Documentation.* https://docs.github.com/en/actions

- Redis Labs. (2024). *Redis Documentation.* https://redis.io/docs/

- Nginx, Inc. (2024). *NGINX Documentation.* https://nginx.org/en/docs/

Ciberseguridad y programación segura

- OWASP Foundation. (2021). *OWASP Top Ten Web Application Security Risks – 2021.* https://owasp.org/www-project-top-ten/

- Python Software Foundation. (2024). *PEP 8 – Style Guide for Python Code.* https://peps.python.org/pep-0008/

- SonarSource SA. (2024). *SonarQube Documentation.* https://docs.sonarsource.com/

- Red Hat. (2023). *Hardening Guide for Linux Servers.* https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/

- Snyk Ltd. (2023). *Bandit Security Analyzer for Python.* https://bandit.readthedocs.io/

Normativas chilenas aplicables

- Biblioteca del Congreso Nacional de Chile. (1999). *Ley N.º 19.628 sobre protección de la vida privada.* https://www.bcn.cl/leychile/navegar?idNorma=141599

- Biblioteca del Congreso Nacional de Chile. (2023). *Ley Marco N.º 21.663 de Ciberseguridad e Infraestructura Crítica de la Información.* https://www.bcn.cl/leychile/navegar?idNorma=1219081

- Ministerio de Hacienda. (2023). *Decreto Supremo N.º 7/2023: Norma Técnica de Seguridad de la Información.* https://www.bcn.cl/leychile/navegar?idNorma=1192704

Bases de datos y modelamiento

- Jiménez Capel, M. Y. (2014). *Bases de datos relacionales y modelado de datos (UF1471).* IC Editorial.

- Coronel, C., & Morris, S. (2023). *Database Systems: Design, Implementation, & Management* (14th ed.). Cengage Learning.

Pruebas de software y QA

- Hambling, B., & Samaroo, P. (2019). *Software Testing: An ISTQB-BCS Certified Tester Foundation Guide* (4th ed.). BCS Learning & Development.

- Django Software Foundation. (2024). *Testing in Django.* https://docs.djangoproject.com/en/5.0/topics/testing/

- Locust.io. (2024). *Locust Load Testing Framework.* https://locust.io/

- SeleniumHQ. (2024). *Selenium WebDriver Documentation.* https://www.selenium.dev/documentation/

Desarrollo y DevOps

- Kim, G., Behr, K., & Spafford, G. (2016). *The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win.* IT Revolution.

- Kim, G., Humble, J., Debois, P., & Willis, J. (2018). *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations.* IT Revolution.