Vicerrectoría Académica
Dirección de Servicios Académicos
Subdirección de Servicios a Escuelas

Sistema de Entrada de Vehículos a Chile (Aduana)
(DAS) Documento Arquitectura de Software
Versión 1.0

Identificación de Documento

Identificación	
Proyecto	Sistema de Entrada de Vehículos a Chile (Aduana)
Versión	1.0

Documento mantenido por	
Fecha de última revisión	
Fecha de próxima revisión	

Documento aprobado por	
Fecha de última aprobación	

Historia de Revisiones

Fecha	Versión	Descripción	Autor

Tabla de Contenidos

1. INTRODUCCIÓN

1.1. Contexto del Problema (General)

La Aduana chilena enfrenta problemas críticos al momento del ingreso de vehículos en pasos fronterizos. Las largas filas, la validación manual de documentos y la falta de interoperabilidad entre organismos como SAG, PDI y Registro Civil provocan demoras, riesgos de error humano y mal uso de recursos. Se requiere modernizar el sistema para asegurar eficiencia, trazabilidad y seguridad.

1.2. Propósito

El propósito de este proyecto es diseñar un sistema informático web que permita a los ciudadanos declarar previamente su ingreso al país con vehículos, productos o mascotas, y que al mismo tiempo permita a los técnicos aduaneros validar dicha información de forma digital e integrada con servicios externos del Estado.

1.3. Ámbito

El sistema está enfocado exclusivamente en el ingreso terrestre de vehículos a Chile. No contempla salida ni tránsito, ni validación de personas. Incluye funcionalidades para ciudadanos, técnicos aduaneros y jefes operativos, abarcando los módulos de validación de vehículos, productos no equipaje, mascotas (perros/gatos), e informes estadísticos.

1.4. Definiciones, acrónimos y abreviaciones

ACRONIMO	DESCRIPCION
CZE	Certificado Zoosanitario de Exportación, requerido para ingreso de mascotas
SAG	Servicio Agrícola y Ganadero, entidad que regula el ingreso de productos y animales
PDI	Policía de Investigaciones, entidad que verifica autorización de ingreso
API	Application Programming Interface, permite integración con servicios externos
MVC	Modelo Vista Controlador, patrón arquitectónico usado en el sistema
DAO	Data Access Object, patrón para separar lógica de acceso a datos
BD	Base de Datos, sistema de almacenamiento de la información del sistema

1.5. Resumen ejecutivo (General)

El proyecto busca mejorar la eficiencia en el proceso de ingreso de vehículos al país mediante un sistema informático que digitaliza el formulario de declaración, automatiza validaciones documentales y genera informes. Utiliza tecnologías web, arquitectura modular, y se integra con APIs externas como Registro Civil, SAG y PDI.

1.6. Arquitectura del sistema (General)

(ej. vista de escenario, vista lógica, vista de desarrollo, vista de proceso, vista física)

La arquitectura se basa en el modelo 4+1 de vistas:

- Vista de Escenario: Modela los casos de uso del sistema, centrados en el ciudadano y el técnico aduanero.
- Vista Lógica: Representa las entidades del sistema y sus relaciones (clases como Ciudadano, Vehículo, Producto, etc.).
- Vista de Desarrollo: Muestra cómo está organizado el código en paquetes y componentes reutilizables.
- Vista de Proceso: Describe el flujo de actividades desde el ingreso de datos hasta la validación y notificación del estado.
- Vista Física: Detalla cómo se despliega el sistema en servidores, con integración a servicios externos.

2. VISIÓN DEL SISTEMA (General)

2.1. Descripción general del sistema

El sistema está diseñado como una plataforma web que permite a los ciudadanos que ingresan a Chile declarar previamente los vehículos, productos no considerados equipaje y mascotas que llevarán consigo. El sistema procesa esta información, valida los documentos mediante integración con APIs externas y entrega una interfaz para que los técnicos de aduana puedan verificar los antecedentes y aprobar o rechazar el ingreso. También incluye una interfaz para que el jefe operativo genere informes estadísticos sobre los ingresos registrados.

2.2. Objetivos del sistema

- Disminuir los tiempos de espera en los pasos fronterizos.
- Mejorar la trazabilidad y control documental de vehículos, productos y mascotas que ingresan al país.
- Reducir el trabajo manual del personal aduanero mediante automatización de procesos.
- Integrar a través de servicios web las validaciones con Registro Civil, PDI y SAG.

2.3. Principales funcionalidades esperadas

- Formulario de ingreso de vehículos (con o sin vehículo de arrastre).
- Subida de documentación obligatoria (padrón, seguro internacional, poder notarial).
- Declaración de productos no equipaje y validación con SAG.
- Adjuntar y validar el certificado CZE para mascotas (perros y gatos).
- Interfaz de revisión técnica para validar cada ingreso.
- Generación automática de informes estadísticos para el jefe operativo.

2.4. Supuestos y dependencias

- El ciudadano debe contar con Clave Única para autenticarse en el sistema.
- Toda la documentación cargada debe tener una antigüedad máxima de 10 días.

- El sistema se conecta en tiempo real con servicios externos: Registro Civil (validación de datos vehiculares), PDI (autorizaciones de ingreso), y SAG (productos y mascotas).

- Se asume que los dispositivos utilizados en frontera tienen conexión estable a internet.

3. ESTILOS Y PATRONES ARQUITECTÓNICOS (General)

3.1. Estilo arquitectónico adoptado (ej. monolítico, microservicios, SOA, capas)

El sistema utiliza un estilo en capas (n-tier) combinado con el patrón Modelo-Vista-Controlador (MVC). Esta elección permite separar de manera lógica la interfaz de usuario (presentación), la lógica de negocio (controladores y validaciones) y el acceso a datos (modelo y DAO). También se estructura como una arquitectura modular, donde cada componente maneja una funcionalidad específica.

3.2. Justificación del estilo según el contexto del sistema

El enfoque en capas facilita la escalabilidad del sistema, ya que cada capa puede modificarse, mantenerse o escalarse de forma independiente. El uso de MVC permite desarrollar una plataforma web clara, organizada y de fácil mantenimiento. Dado que el sistema interactúa con múltiples actores (ciudadano, técnico, jefe operativo) y con servicios externos, esta arquitectura garantiza separación de responsabilidades, reutilización de componentes y desacoplamiento entre vistas y lógica de negocio.

3.3. Patrones de diseño aplicados (ej. patrón MVC, repositorio, etc.)

- MVC (Modelo-Vista-Controlador): estructura principal para separar lógica de presentación, lógica de control y modelo de datos.

- DAO (Data Access Object): patrón para el acceso a datos, separando consultas SQL del resto de la lógica.

- Fachada (Facade): utilizado para encapsular la comunicación con APIs externas (Registro Civil, PDI, SAG), simplificando su integración y desacoplando el sistema de su implementación.

- Singleton (opcional para servicios de acceso único como conexión a base de datos).

4. MODELO 4 +1 Y VISTAS ARQUITECTÓNICAS

4.1. VISTA DE ESCENARIO (General y salida vehículo **o** entrada vehículo)

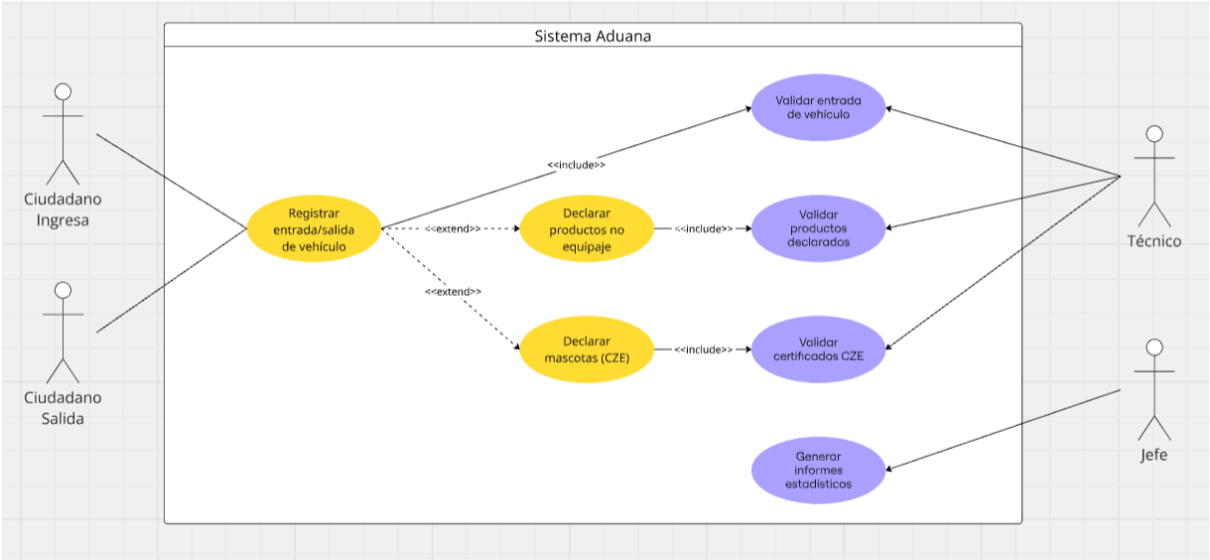
4.1.1. Propósito (General)

Representar las interacciones entre los usuarios del sistema (ciudadano, técnico en revisión y jefe operativo) con los principales procesos funcionales relacionados con el ingreso de vehículos a Chile. Esta vista permite comprender el alcance del sistema y delimitar los roles y funcionalidades.

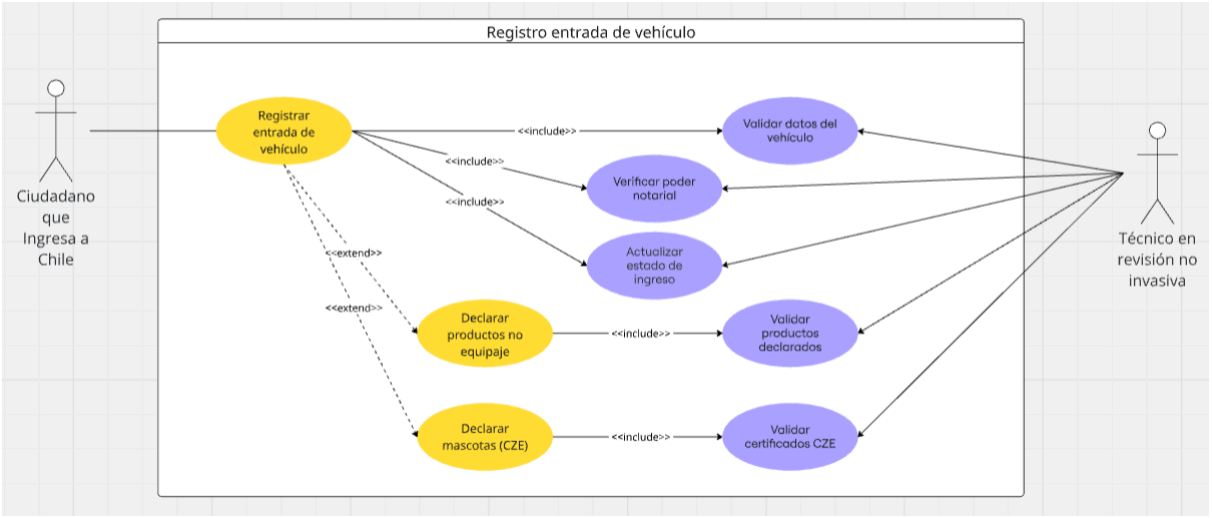
4.1.2. Actores (General)

- Ciudadano que ingresa a Chile: Declara información del vehículo, productos y mascotas mediante formulario web.
- Técnico en revisión no invasiva: Valida los datos ingresados por el ciudadano y decide si se aprueba o rechaza el ingreso.
- Jefe Operativo: Genera informes estadísticos a partir de los registros procesados.

4.1.3. Diagrama general de casos de uso (General)



4.1.4. Diagrama de casos de uso específicos (salida vehículo **o** entrada vehículo)



4.1.5. Lista de casos de uso (salida vehículo o entrada vehículo)

Código	Nombre	Actores
CU-001-001	Registrar entrada de vehículo	Ciudadano, Técnico
CU-001-002	Declarar productos no equipaje	Ciudadano, Técnico
CU-001-003	Declarar mascotas con CZE	Ciudadano, Técnico
CU-001-004	Validar documentos del vehículo	Técnico
CU-001-005	Generar informes estadísticos	Jefe Operativo

4.1.6. Especificación de casos de uso (**UN** caso de uso principal de la salida vehículo/entrada vehículo)

Caso de Uso	Registrar entrada de vehículo	**Identificador:** CU-001-001
Actores	Ciudadano, Técnico	
Tipo	Primario	
Referencias	Ley de Aduanas, validación PDI, validación Registro Civil	
Precondición	El ciudadano debe estar autenticado y contar con Clave Única	
Postcondición	El sistema guarda la solicitud con su estado inicial como “Pendiente de validación”	
Descripción	El ciudadano declara los datos del vehículo, sube la documentación correspondiente y completa la información del viaje	
Resumen	Este caso permite registrar el vehículo antes de cruzar la frontera, enviando los datos al sistema para su validación técnica posterior	

CURSO NORMAL

Nro.	Ejecutor	Paso o Actividad
1	Ciudadano	Ingresa al sistema con Clave Única.
2	Sistema	Valida las credenciales del usuario.
3	Ciudadano	Completa formulario con datos del vehículo, conductor y viaje.
4	Ciudadano	Adjunta padrón, seguro internacional, y poder notarial si aplica.
5	Sistema	Verifica que los documentos no superen los 60 días de antigüedad.
6	Sistema	Registra el ingreso como solicitud “Pendiente de revisión técnica”.
7	Técnico	Accede al formulario y visualiza la solicitud.
8	Técnico	Valida documentos e información declarada.
9	Técnico	Cambia el estado a Aceptado, Pendiente o Rechazado.

CURSO ALTERNATIVO

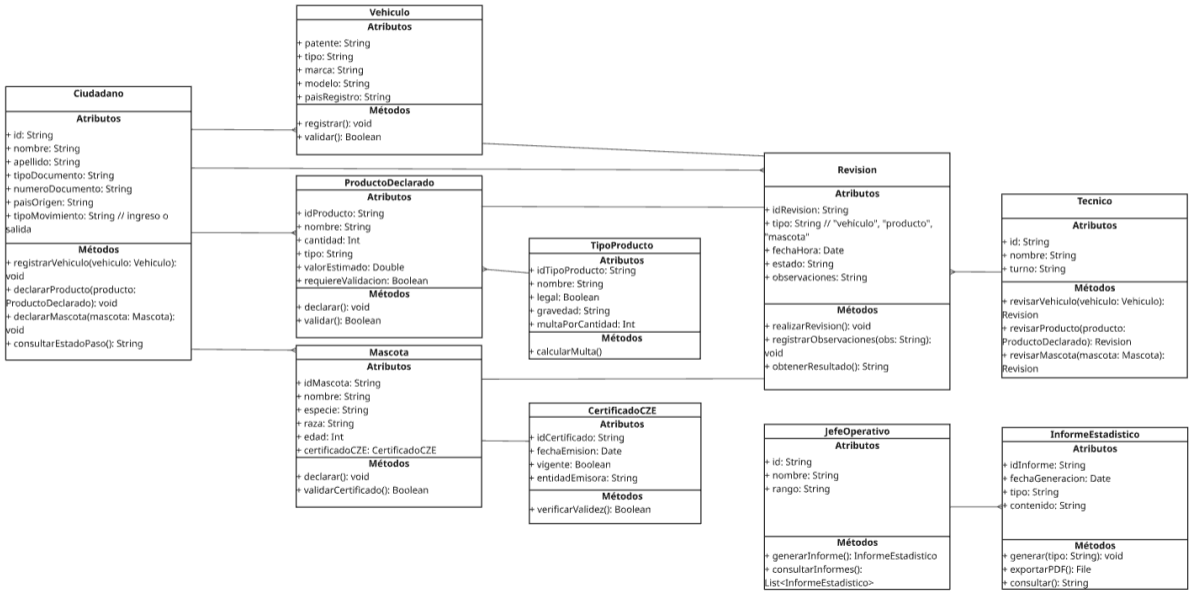
Nro.	Descripción de acciones alternas
4	Si el ciudadano no cuenta con el poder notarial, el sistema marca la solicitud como Incompleta.
5	Si el documento tiene más de 60 días, el sistema bloquea el envío y solicita uno actualizado.
8	Si el técnico detecta inconsistencias, puede dejar observaciones y marcar como “Pendiente”.

4.2. VISTA LÓGICA (salida vehículo **o** entrada vehículo)

4.2.1. Propósito

La vista lógica tiene como objetivo representar la estructura interna del sistema mediante clases, atributos, métodos y relaciones entre objetos. Esto permite visualizar cómo se organizan los datos, cómo interactúan las entidades del dominio y cómo se construye la base para el desarrollo orientado a objetos.

4.2.2. Diagrama de clases



4.2.3. Descripción diagrama de clases

El sistema se estructura a través de entidades representativas del dominio aduanero:

- Ciudadano se relaciona con los objetos Vehículo, ProductoDeclarado y Mascota.
- Mascota está asociada a un CertificadoCZE, que debe validarse antes del ingreso.
- Técnico ejecuta Revisión sobre los objetos declarados, que pueden incluir observaciones y resultados.
- InformeEstadístico es generado por el JefeOperativo y agrupa datos históricos relevantes.

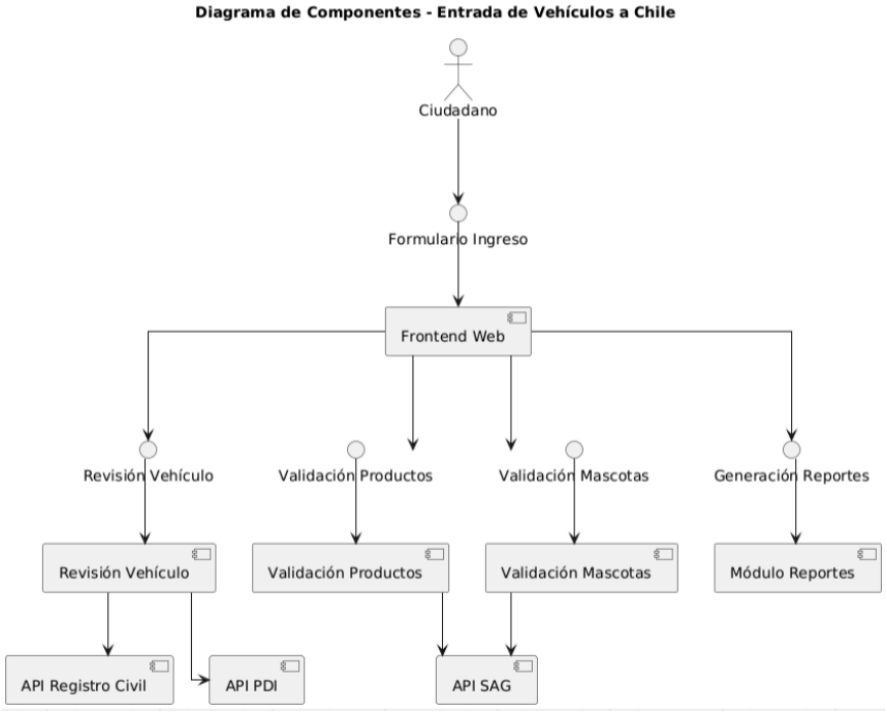
Cada clase posee sus atributos principales y métodos clave como validar(), declarar(), generarInforme(), siguiendo principios de cohesión, encapsulamiento y claridad funcional.

4.3. VISTA DE IMPLEMENTACIÓN/DESARROLLO (salida vehículo o entrada vehículo)

4.3.1. Propósito

Representar la organización física del software a través de sus componentes funcionales. Permite modelar cómo se dividen los módulos del sistema y cómo se comunican a través de interfaces, facilitando el trabajo modular y la reutilización.

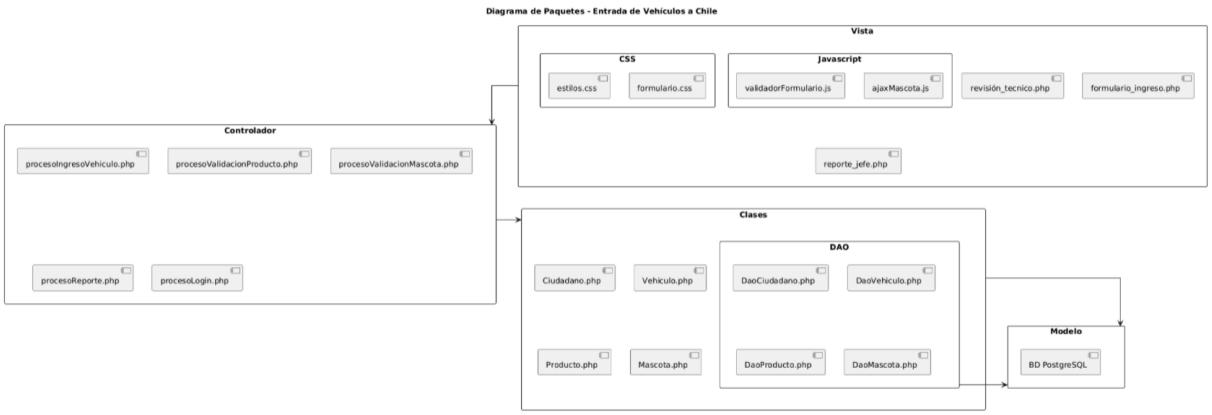
4.3.2. Diagrama de componente



4.3.3. Descripción diagrama de componente

Cada módulo cumple una función aislada y se comunica mediante interfaces claramente definidas. El componente “Frontend Web” orquesta la interacción con el usuario. Las validaciones están distribuidas en componentes especializados según el tipo de entidad (producto, mascota, vehículo). La integración con APIs externas está encapsulada por componentes fachada para desacoplar el sistema interno de las implementaciones externas.

4.3.4. Diagrama de paquete



4.3.5. Descripción diagrama de paquete

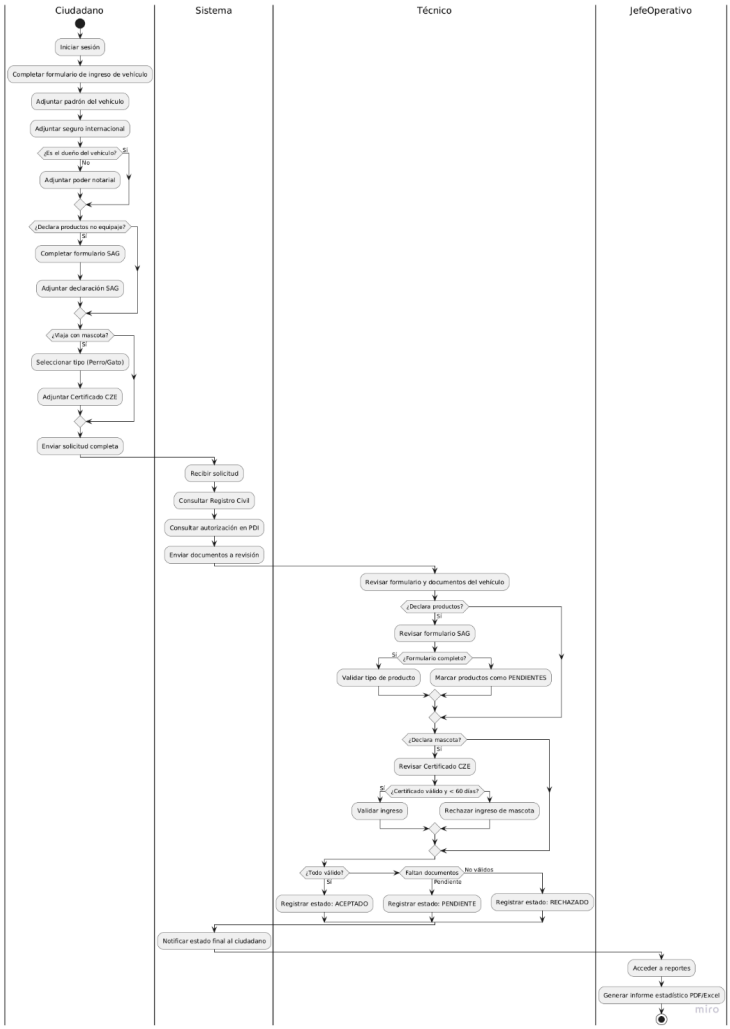
El sistema está organizado de forma modular, siguiendo una estructura MVC extendida. Los paquetes están claramente separados por función y nivel de abstracción. Esto permite una distribución eficiente del código y facilita el mantenimiento, escalabilidad y la incorporación futura de nuevas funcionalidades.

4.4. VISTA DE PROCESOS (salida vehículo **o** entrada vehículo)

4.4.1. Propósito

Modelar el flujo dinámico del sistema. Esta vista permite identificar el orden en que ocurren las actividades, la participación de cada actor, las decisiones condicionales y la secuencia de validaciones que atraviesa cada declaración.

4.4.2. Diagrama de actividad



4.4.3. Descripción diagrama de actividad

El diagrama inicia con la autenticación del ciudadano, seguido por el llenado del formulario y la carga de documentos. Luego, el sistema valida los plazos y estructura la información como solicitud pendiente. El técnico revisa los datos, verifica la autenticidad documental y emite una resolución. Si hay productos o mascotas, se activan subflujos con sus respectivas validaciones. Finalmente, se actualiza el estado de la solicitud.

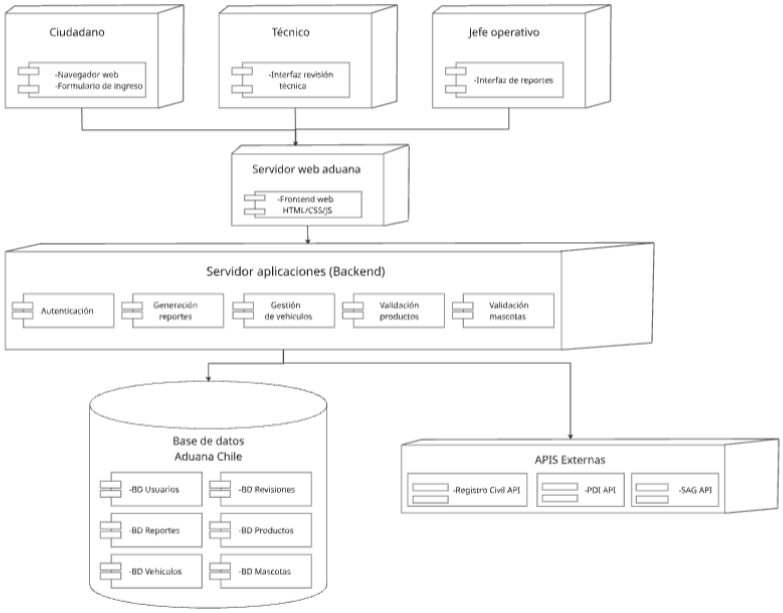
4.5. VISTA FÍSICA (salida vehículo **o** entrada vehículo)

4.5.1. Propósito

La vista física tiene como finalidad mostrar cómo se despliega el sistema en una infraestructura tecnológica concreta, identificando los distintos nodos, servidores y conexiones entre componentes físicos y lógicos. Esta vista es esencial para comprender la distribución real del sistema, su escalabilidad, rendimiento y puntos

de integración con servicios externos.

4.5.2. Diagrama de despliegue



4.5.3. Descripción diagrama de despliegue

El sistema se ejecuta como una aplicación web distribuida. El ciudadano accede al formulario desde cualquier dispositivo conectado, que se comunica con el servidor web. Este redirige las peticiones al backend, donde se ejecutan los procesos de validación y lógica de negocio.

La base de datos almacena de forma segura todas las entidades del sistema, mientras que la comunicación con APIs externas permite validar documentos oficiales (como el padrón del vehículo o el certificado CZE) sin intervención manual.

Además, el técnico aduanero y el jefe operativo acceden desde terminales internos conectados a la misma aplicación, pero con interfaces específicas y control de roles.

5. REQUISITOS DE CALIDAD (General)

5.1. Propósito

Esta sección define los atributos de calidad esperados del sistema, en línea con el modelo ISO/IEC 25010. Establece cómo se medirá y garantizará la calidad del producto software, considerando factores como usabilidad, rendimiento, seguridad, mantenibilidad, entre otros.

5.2. Atributos de calidad (por ejemplo: Usabilidad, Accesibilidad (WCAG), Rendimiento, Mantenibilidad, Seguridad Portabilidad)

ATRIBUTO DE CALIDAD	DESCRIPCIÓN	JUSTIFICACIÓN
Usabilidad	El sistema debe ser intuitivo, fácil de usar y comprensible para cualquier usuario.	El sistema será utilizado por ciudadanos comunes, por lo tanto debe minimizar la curva de aprendizaje.
Rendimiento	El tiempo de respuesta debe ser inferior a 2 segundos en operaciones clave.	Las validaciones deben ser rápidas para evitar demoras y acumulación de personas en frontera.
Seguridad	Se requiere autenticación, validación de usuarios y protección de datos.	El sistema trata con datos personales sensibles (RUT, certificados, etc.), por lo que debe ser seguro.
Mantenibilidad	La arquitectura debe permitir actualizaciones y corrección de errores fácilmente.	El sistema puede evolucionar (nuevas reglas o validaciones), por lo que debe ser fácilmente modificable.
Portabilidad	El sistema debe ser accesible desde distintos dispositivos (PC y móviles).	Se busca que el ciudadano pueda acceder al formulario desde su hogar o celular antes de viajar.

5.3. Reglas y criterios de evaluación de calidad

Cómo se medirá el cumplimiento de cada atributo (ej. tiempo de carga < 2 seg, puntuación Nielsen de usabilidad, cumplimiento WCAG nivel AA, etc.).

Herramientas o métodos que se utilizarán(pruebas de carga, pruebas heurísticas, inspecciones, validación con usuarios, etc)

- Usabilidad: Evaluada mediante pruebas heurísticas y validación con usuarios. Se aplicarán principios de Nielsen para verificar claridad, consistencia y visibilidad del estado del sistema.

- Rendimiento: Se realizarán pruebas de carga con simulación de múltiples solicitudes simultáneas. Se medirá el tiempo de respuesta, el tiempo de procesamiento de formularios y generación de reportes.

- Seguridad: Se asegurará que todas las conexiones sean HTTPS, los documentos estén protegidos, y el acceso esté restringido por rol. Se evaluará mediante pruebas de vulnerabilidad y control de acceso.

- Mantenibilidad: Se verificará la modularidad del código, el cumplimiento de principios SOLID, y la existencia de documentación técnica.

- Portabilidad: Se harán pruebas de compatibilidad en navegadores modernos (Chrome, Firefox, Edge) y en dispositivos móviles Android/iOS.

6. PRINCIPIOS DE DISEÑO APLICADOS

6.1. Propósito

El objetivo de esta sección es evidenciar cómo se aplicaron principios fundamentales de la Ingeniería de Software en el diseño del sistema, con el fin de lograr una arquitectura robusta, mantenible y de calidad, que permita escalar el sistema y adaptarse a cambios futuros.

6.2. Principios de diseño

PRINCIPIO	DESCRIPCIÓN	APLICACIÓN EN EL SISTEMA
Cohesión	Cada módulo o clase tiene una única responsabilidad bien definida.	Los servicios están diseñados para realizar tareas específicas y no múltiples funciones
Bajo acoplamiento	Las clases y componentes están lo menos dependientes entre sí.	Los módulos (vehículos, productos, mascotas) están desacoplados y se comunican por interfaces.
Encapsulamiento	La lógica interna de cada clase está protegida del acceso externo directo.	La validación y manipulación de datos se hace a través de métodos definidos, protegiendo los atributos.
Modularidad	El sistema está dividido en componentes reutilizables y autónomos.	Las funcionalidades están separadas por paquetes y componentes: autenticación, revisión, reportes, etc.
Abstracción	Se ocultan detalles internos y se exponen solo funcionalidades relevantes.	Se aplican interfaces para declarar mascotas, productos y vehículos sin revelar la implementación técnica.

7. PROTOTIPO

7.1. Propósito

El propósito del prototipo es representar de manera visual y funcional la interfaz del sistema antes de su implementación definitiva. A través del prototipado, se pueden validar ideas de diseño, identificar posibles problemas de usabilidad y obtener retroalimentación temprana, lo que reduce riesgos y costos en etapas posteriores del desarrollo. Este proceso permite simular la interacción del usuario con el sistema y evaluar la estructura, el flujo y la estética de las pantallas propuestas.

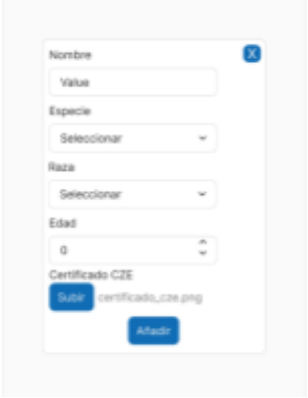
7.2. Mockups (imágenes con una breve descripción)

1. Menú principal: Pantalla de inicio donde el usuario puede elegir entre registrar una nueva solicitud de ingreso o consultar el estado de solicitudes anteriores.

2. Formulario de Ingreso: Formulario inicial donde el usuario ingresa sus datos personales como RUN, número de documento, nombre, apellido y país de origen.

3. Archivos y declaraciones: Sección donde el usuario puede subir documentos requeridos, como el padrón del vehículo, acceder al formulario SAG y registrar mascotas que viajan.

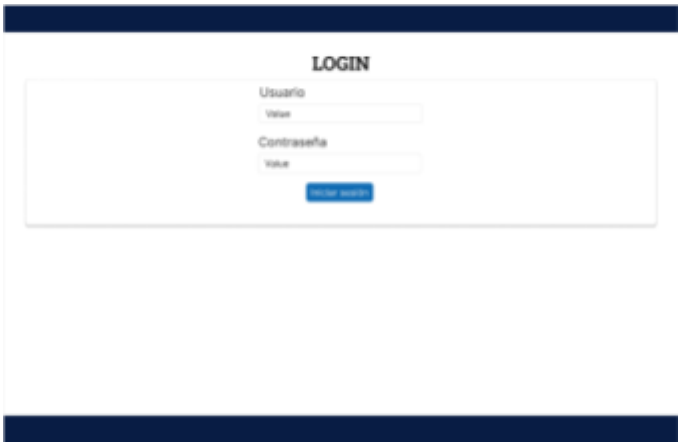
4. Registro de mascota: Formulario específico para ingresar información de mascotas, incluyendo nombre, especie, raza, edad y certificado CZE.



5. Confirmación de envío: Pantalla que informa al usuario que su solicitud fue enviada correctamente, permitiéndole continuar con el proceso.

6. Consulta de solicitudes: Vista que muestra el historial de solicitudes ingresadas por el usuario, incluyendo su fecha, hora y estado actual (aceptado, pendiente o denegado).

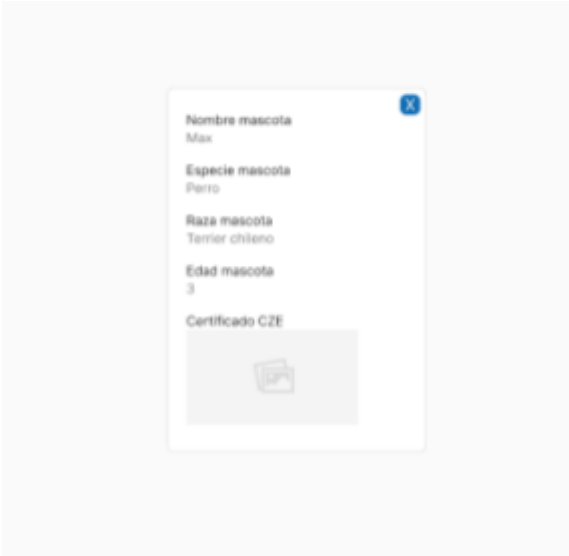
7. Login: Pantalla de inicio de sesión donde el usuario debe ingresar su nombre de usuario y contraseña para acceder al sistema.



8. Listado de Solicitudes: Vista principal del sistema tras iniciar sesión, donde se muestran todas las solicitudes realizadas. Cada registro incluye el RUN del solicitante, su nombre, fecha y hora de solicitud, estado actual (Aceptado, Denegado o Pendiente) y un botón para visualizar más detalles.

9. Detalle de Solicitud: Pantalla que muestra toda la información asociada a una solicitud específica: ID, RUN, nombre y apellido del solicitante, país de origen, imagen del padrón del vehículo, botón para ver el formulario del SAG, listado de mascotas asociadas (con botón para ver sus detalles), estado de la solicitud y botones para Aceptar, Marcar como Pendiente o Denegar.

10. Detalle de Mascota: Ventana emergente que despliega los datos completos de una mascota: nombre, especie, raza, edad y espacio para cargar o visualizar el certificado CZE.



7.3. Justificar herramientas de prototipado

Se utilizó Figma como herramienta principal para el diseño de prototipos debido a su capacidad de trabajo colaborativo en línea, su interfaz intuitiva y sus potentes funciones de diseño y prototipado interactivo. Figma permite crear mockups de alta fidelidad, compartirlos fácilmente con otros miembros del equipo y recibir comentarios en tiempo real. Además, su enfoque basado en la nube evita problemas de compatibilidad y facilita la iteración constante en el diseño.

8. EVALUACIÓN DE CALIDAD HEURÍSTICA DE NIELSEN

8.1. Propósito

El propósito de esta evaluación es identificar fortalezas y debilidades en la interfaz del sistema, aplicando los diez principios heurísticos de usabilidad definidos por Jakob Nielsen. Esta metodología permite detectar problemas potenciales que podrían afectar la experiencia del usuario, estimar su nivel de severidad y proponer recomendaciones concretas de mejora. Su aplicación contribuye a asegurar una interacción eficiente, intuitiva y satisfactoria con el sistema, incluso antes de realizar pruebas con usuarios reales.

8.2. Lista de verificación

Para llevar a cabo la evaluación, se utilizó una tabla de verificación basada en los principios de Nielsen. En ella, cada criterio fue evaluado considerando si se cumple (✔) o no (✗), acompañado de observaciones que justifican la decisión y un nivel de gravedad estimado para cada posible problema identificado. Esta lista permite una revisión sistemática y estructurada de la interfaz, asegurando que se consideren todos los aspectos fundamentales de la usabilidad.

Nº	Principio de Usabilidad de Nielsen	Criterio de Evaluación	¿Se cumple? (✓/✗)	Observaciones / Evidencia	Gravedad del problema
1	**Visibilidad del estado del sistema**	¿El sistema informa claramente al usuario de lo que está ocurriendo (cargas, acciones)?	✓	El sistema notifica cuando una solicitud ha sido enviada y aprobada.	Baja
2	**Correspondencia entre el sistema y el mundo real**	¿La terminología y flujos se relacionan con el lenguaje y lógica del usuario?	✓	La terminología utilizada (vehículo, revisión, mascota) se adapta al lenguaje del usuario común.	Baja
3	**Control y libertad del usuario**	¿El usuario puede deshacer/repetir acciones fácilmente?	✓	El usuario puede modificar entradas antes del envío definitivo y cancelar operaciones.	Baja
4	**Consistencia y estándares**	¿Se mantiene un diseño coherente entre pantallas, botones y mensajes?	✓	Se mantiene un diseño visual coherente en todas las interfaces del sistema web.	Baja
5	**Prevención de errores**	¿El diseño evita que ocurran errores antes de que sucedan?	✓	El sistema valida campos obligatorios y formatos antes del envío para prevenir errores.	Baja
6	**Reconocimiento mejor que recuerdo**	¿Las opciones y funciones son visibles sin que el usuario deba recordar información?	✓	El sistema muestra menús desplegables y opciones visibles sin tener que memorizar códigos.	Baja
7	**Flexibilidad y eficiencia de uso**	¿Permite atajos o personalización para usuarios avanzados?	✓	El sistema entrega ayuda contextual como íconos informativos y textos de apoyo.	Baja
8	**Diseño estético y minimalista**	¿La interfaz evita información innecesaria o ruido visual?	✓	Interfaz limpia con distribución clara de elementos. No hay elementos visuales innecesarios.	Baja
9	**Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores**	¿Los mensajes de error son claros, comprensibles y ofrecen solución?	✓	El sistema muestra mensajes de error comprensibles y permite retomar el flujo sin pérdida de información.	Baja
10	**Ayuda y documentación**	¿Existe ayuda accesible, clara y orientada a la tarea cuando el usuario lo necesita?	✓	Los manuales están disponibles en línea y la interfaz es autoexplicativa, guiando al usuario sin requerir soporte adicional.	Baja

8.3. Análisis y métricas de resultados

La evaluación heurística realizada con base en los 10 principios de usabilidad de Nielsen permitió identificar que la interfaz del sistema cumple satisfactoriamente con todos los criterios evaluados. Cada principio fue examinado mediante preguntas clave, observaciones de la interfaz y una categorización de la gravedad de los

posibles problemas detectados. A continuación, se resumen los hallazgos principales:

- Cumplimiento total: Se verificó el cumplimiento de los 10 principios heurísticos (✓), lo cual indica que el sistema está diseñado con una sólida orientación a la usabilidad, brindando una experiencia amigable, coherente y predecible para los usuarios.

- Tipo y gravedad de los problemas: Todas las observaciones correspondieron a aspectos de gravedad baja, sin identificar fallos críticos ni obstáculos relevantes para la realización de tareas. Esto sugiere que los problemas detectados no afectan significativamente la eficiencia, efectividad ni satisfacción del usuario, pero sí representan oportunidades de mejora.

- Áreas de mejora observadas:

    - Incrementar la personalización y eficiencia para usuarios avanzados (por ejemplo, atajos de teclado o configuraciones personalizables).

    - Reforzar la claridad en los mensajes de error, haciendo énfasis en soluciones sugeridas cuando corresponda.

    - Ofrecer más guías interactivas o mensajes contextuales que acompañen procesos menos intuitivos.

- Métrica cualitativa general: La interfaz demuestra una alta alineación con las buenas prácticas de usabilidad, y destaca por su consistencia visual, claridad terminológica y prevención de errores mediante validaciones apropiadas.

9. CONTROL DE VERSIONES

9.1. Propósito

El propósito del control de versiones en un proyecto de software es llevar un seguimiento organizado y estructurado de los cambios realizados a lo largo del desarrollo. Permite identificar con precisión qué modificaciones se han efectuado, cuándo se hicieron, quién las realizó y por qué. Esta práctica es fundamental para mantener la integridad del proyecto, facilitar el trabajo colaborativo, recuperar versiones anteriores en caso de errores y documentar la evolución del sistema.

9.2. Control de versión utilizado (justificar el tipo de control de versión utilizad (fecha, semántica o secuencial)

Para este proyecto se utilizó un control de versión semántico, basado en el esquema MAJOR.MINOR.PATCH, donde cada número refleja un tipo de cambio específico: cambios mayores que afectan la compatibilidad, mejoras menores que agregan funcionalidades sin romper lo existente, y parches que corrigen errores. Este

sistema fue elegido por su claridad y por ser ampliamente adoptado en proyectos de desarrollo, lo cual facilita la comprensión de las actualizaciones tanto para el equipo como para terceros.

9.3. Justificar herramientas de versionamiento

Se utilizó Git como herramienta de control de versiones, debido a su eficacia, flexibilidad y capacidad para trabajar de forma distribuida. Git permite llevar un historial detallado de los cambios, facilitar la colaboración entre distintos integrantes del equipo, gestionar ramas para el desarrollo paralelo y resolver conflictos de manera eficiente. La herramienta fue utilizada en conjunto con la plataforma GitHub, que proporciona una interfaz amigable para la gestión de repositorios, seguimiento de incidencias y revisión de código, optimizando así el flujo de trabajo y asegurando la trazabilidad del desarrollo.

10. CONCLUSIONES

El desarrollo del sistema para la entrada de vehículos a Chile demuestra cómo la Ingeniería de Software, apoyada en estándares y buenas prácticas de modelamiento UML, puede resolver problemas reales de eficiencia y control en procesos críticos como el aduanero. Mediante el uso de diagramas de casos de uso, clases, componentes, paquetes, despliegue y actividades, fue posible representar con claridad tanto la funcionalidad esperada como la estructura interna y externa del sistema.

Se logró una arquitectura modular, segura y fácilmente mantenible, que permite integrar servicios externos (como SAG, PDI y Registro Civil) y mejorar la experiencia del usuario final. Asimismo, se aplicaron principios sólidos de diseño (como cohesión, bajo acoplamiento y modularidad), y se definieron atributos de calidad que garantizan usabilidad, rendimiento y confiabilidad.

Además, se diseñó una interfaz intuitiva y responsiva, que contempla pantallas como login, listado de solicitudes y visualización detallada de cada solicitud, incluyendo los antecedentes del solicitante, sus mascotas, el padrón del vehículo y los formularios requeridos. Este diseño busca facilitar el trabajo de los funcionarios y reducir errores administrativos.

La incorporación de enlaces al formulario SAG y la visualización directa de certificados (como el CZE para mascotas) permiten una verificación rápida y eficiente de los requisitos exigidos por normativa, agilizando la toma de decisiones por parte del personal autorizado.

Finalmente, el enfoque adoptado promueve una solución escalable, que podría extenderse a otros puntos de control fronterizo o integrarse con nuevas instituciones, fortaleciendo la trazabilidad y transparencia de los procesos asociados al ingreso de personas, vehículos y mascotas al país.

11. BIBLIOGRAFÍA

Pressman, R. S. (2010). *Ingeniería del Software. Un enfoque práctico* (7ma Ed.). McGraw-Hill.

ISO/IEC 25010:2011. *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE).*

Kruchten, P. (1995). *Architectural Blueprints—The "4+1" View Model of Software Architecture.*

Documentación oficial de SAG, PDI y Registro Civil de Chile.

12. ANEXOS

12.1. Repositorio GitHub del Proyecto

Durante el desarrollo del sistema, se utilizó un repositorio en GitHub para almacenar los documentos generados a lo largo del semestre. Esto permitió mantener un control de versiones y asegurar la trazabilidad del trabajo.
Enlace al repositorio: https://github.com/MaiSaldivia/ing_software_aduanas

12.2. Captura de pantalla del repositorio

A continuación, se muestra una imagen del repositorio en GitHub donde se puede visualizar la estructura de carpetas y archivos subidos: