APLICACIÓN WEB VIAL PLUS PARA LA OPERACIÓN DE PEAJES, MÓDULOS GESTIÓN DE USUARIOS Y CONFIGURACIÓN DE ESTACIÓN

ANTONIO ADOLFO MATHEUS ZABALA

UNIVERSIDAD ANTONIO NARIÑO

FACULTAD DE INGENIERÍA DE SISTEMAS

ESPECIALIZACIÓN INGENIERÍA DE SOFTWARE

BOGOTÁ D.C.

2020

APLICACIÓN WEB VIAL PLUS PARA LA OPERACIÓN DE PEAJES, MÓDULOS GESTIÓN DE USUARIOS Y CONFIGURACIÓN DE ESTACIÓN

ANTONIO ADOLFO MATHEUS ZABALA

Proyecto de grado para optar al título de Especialista en Ingeniería de Software

Asesores

ING. DIANALIN NEME

Directora Programa

ING. IVÁN ROMERO

Docente Académico

UNIVERSIDAD ANTONIO NARIÑO

FACULTAD DE INGENIERÍA DE SISTEMAS

ESPECIALIZACIÓN INGENIERÍA DE SOFTWARE

BOGOTÁ D.C.

2020

Agradecimientos

A mi esposa y mi familia, los cuales son la motivación para seguir creciendo tanto personalmente como profesionalmente.

A la empresa Thomas Instruments por darme el apoyo en la realización del proyecto educativo en la Especialización de Ingeniería de Software.

A los docentes de la Universidad por compartir sus conocimientos y experiencia en esta etapa de aprendizaje.

Tabla de Contenidos

Lista de tablas

Lista de figuras

APLICACIÓN WEB VIAL PLUS PARA LA OPERACIÓN DE PEAJES, MÓDULOS GESTIÓN DE USUARIOS Y CONFIGURACIÓN DE ESTACIÓN

Resumen

Web Vial es una plataforma existente para el control de recaudo y gestión de operación de peajes, la cual presenta dificultades en todas las etapas de su proceso de producción. El presente proyecto, toma un conjunto de funcionalidades básicas del actual sistema agrupadas en dos módulos con el fin de desarrollar una alternativa denominada 'Web Vial Plus', presentando una solución como herramienta que permita la restructuración de la aplicación.

En el documento se presentan los problemas detectados y el correspondiente proceso que permita obtener una arquitectura de software basada en microservicios aplicando estándares y patrones, apoyado en una metodología de proyectos de acuerdo con las necesidades del proyecto y entregar a la empresa Thomas Instruments las estructuras, herramientas y guías que le permitan desarrollar la aplicación de forma eficiente, orientada a la escalabilidad, extensibilidad, reduciendo el tiempo de lanzamiento de funcionalidades buscando el aumento de calidad comparado con el actual ciclo de producción.

Introducción

En el presente proyecto se establece la problemática que tiene la empresa Thomas Instruments S.A. con su plataforma Web Vial, el cual se ofrece en su portafolio. El producto en mención es uno de los productos que actualmente soporta la compañía para el control de recaudo manual y electrónico de peajes y su gestión de operación, está implementado sobre tecnologías que a la fecha presentan obsolescencia tecnológica, además de no tener soporte de algunos fabricantes.

El desarrollo del producto con dichas tecnologías obliga a que las aplicaciones sean monolíticas en su arquitectura y que sus componentes presenten alto grado de acoplamiento y dificultad para integrar con plataformas externas, lo que genera sobre costos en tiempo y recursos del ciclo de vida normal de construcción de software y dificultades al realizar las actividades de mantenimiento y soporte. Adicionalmente, se identifica que la aplicación ha dejado de crecer detenido su factor de innovación, debido a que los esfuerzos se han dirigido completamente a procesos correctivos.

Por lo anterior, es el momento de redefinir las actividades que se están realizando para la construcción de las aplicaciones de la compañía, basados en conceptos de ingeniería de software. Se establecerá una arquitectura que permita mejorar las características del producto actual unido a procesos y herramientas actualizadas incrementando el valor tanto hacia el lado del cliente (calidad) como interno (costos).

El desarrollo del proyecto se realizará a lo largo del año 2020, estableciendo que en el mes de noviembre se entregue una solución a la empresa Thomas Instruments que consta de una aplicación base implementando dos módulos y la correspondiente arquitectura de software planteada para permitir la implementación de las funcionalidades existentes y nuevas, adicionalmente que cumpla con las expectativas de mejora de su producto.

1. Planteamiento del problema

1.1. Descripción del problema

En el actual proceso de construcción de la aplicación Web Vial se detectan los siguientes aspectos como falencias para tener en cuenta, que afectan negativamente la entrega al cliente y puesta en producción.

- Acoplamiento a las siguientes tecnologías de desarrollo obsoletas: Java EE 5. Servidor de aplicaciones para despliegue: Sun GlassFish Enterprise Server v2.1.
- Guías o estándares de desarrollo no existentes: con este aspecto cada desarrollador aplica su criterio y técnicas para llevar a cabo las actividades de inclusión de una nueva funcionalidad o modificación de una existente, sin dejar documentado o especificación del método utilizado. Esto ha llevado al alto acoplamiento y baja cohesión de los componentes y duplicación de código.
- Pruebas funcionales de la aplicación totalmente manuales:
    - Pruebas unitarias no existentes.
    - Pruebas de aceptación no estandarizadas o no existentes.
    - Pruebas de integración mediante procesos manuales.
    - Pruebas de regresión, se llevan a cabo totalmente manuales y la mayoría de los casos no se llega a realizar el set de pruebas completo, debido al tiempo requerido para el proceso (estimado en más de un mes).
- Baja reutilización de componentes y por lo tanto se encuentra duplicación de código.

- Herramientas de automatización no utilizadas. Las librerías o componentes de terceros son obtenidos y referenciados manualmente, lo que dificulta el control y actualización de elementos obsoletos.

1.2. Formulación del problema

El actual producto de software cubre las necesidades de los clientes en la mayoría de las ocasiones, pero con dificultades en su ciclo de vida, tal como lo son las fallas en producción generadas por la baja mantenibilidad de la aplicación, las demoras en la solución de incidentes o entrega de nuevas funcionalidades y nula interoperabilidad con otros sistemas que se ha solicitado en los requisitos de algunos requerimientos.

¿Cómo incrementar la calidad del software, reducir los tiempos de producción y crear procesos que permitan interoperabilidad con plataformas externas generando en los clientes satisfacción y confianza en el producto? El anterior interrogante se establece como la base para el planteamiento de la solución al problema planteado.

1.3. Justificación

La ausencia de procesos claramente definidos y poca organización ha llevado a que el actual software se vuelva obsoleto y muy difícil de mantener, conllevando un aumento de costos, reprocesos, estancamiento de crecimiento e innovación y como factor final, desconfianza de los clientes en la plataforma por falta de calidad y demora en la entrega de las soluciones a sus requerimientos. Por lo anterior, se hace necesario dar un vuelco en el proceso de desarrollo del producto estableciendo una arquitectura para la construcción de la aplicación que permita utilizar métodos y herramientas actualizadas y de esta manera mejorar los aspectos mencionados anteriormente.

1.4. Objetivos

1.4.1. Objetivo General

Desarrollar el Administrador de módulos, el módulo de Gestión y el módulo de Configuración para la aplicación Web Vial Plus con el fin de construir una aplicación escalable.

1.4.2. Objetivos Específicos

Entregar una arquitectura de software como herramienta que sirva como guía para el desarrollo de nuevos módulos para la aplicación Web Vial Plus reflejada en el documento SAD (Software Architecture Document).

Implementar el desarrollo de componentes independientes que permitan modularidad para conjuntos de funcionalidades por módulos y que permitan obtener una alta extensibilidad, escalabilidad y mantenibilidad de la aplicación.

Aplicar estándares de seguridad informática en los módulos de la aplicación basados en la aplicación de controles del Top 10 de Riesgos de Seguridad de OWASP.

Estandarizar los lineamientos de producción del software a nivel de código y base de datos estableciendo guías y recomendaciones de acuerdo con estándares en el SAD.

1.5. Alcance y Limitaciones del Proyecto

El proyecto se limita a establecer una arquitectura aplicada a un par de módulos de la aplicación existente, por lo tanto, la decisión de continuar con su implementación en las demás funcionalidades es de la empresa Thomas Instruments, así como la definición y agrupación de funciones.

Adicionalmente se prepara un proceso que sirva de preparación para posteriormente aplicar metodologías que permitan agilizar aún más el desarrollo de la plataforma, como lo son los procesos de integración y entrega continua.

2. Marco de Referencia.

En el presente capitulo se establecen las necesidades y bases del proyecto, mediante la descripción del estado actual del proceso como lo son las causas de la necesidad del proyecto, alternativas en el mercado y conceptos que ayudarán a plantear la solución del problema.

2.1. Estado del Arte.

En los siguientes apartados se determina el estado actual de los productos que son alternativa o competencia en el mercado de la solución.

2.1.1. Resumen del estado del arte

La empresa Thomas Instruments S.A., hace parte del Grupo Thomas Greg & Sons, organización especializada en productos y servicios de seguridad con más de 50 años de experiencia y con presencia en 12 países alrededor del mundo, especializados en la operación de recaudo de peajes y su control electrónico.

Entre las diferentes opciones de soluciones que se encuentran en el mercado para la operación y administración de peajes se encuentran las siguientes:

Tabla 1

Proveedores de Equipos para Sistemas de Peajes

Empresa	Solución	Origen	Características
DYETRON	PX3	Colombia	Informes y Gráficos: Reportes de la conf. De la estación. Reportes auditoria aplicación. Reportes con la información de vehículos y recaudos de la estación de peaje. Reportes de resúmenes horarios. (Con representación gráfica). Reportes de recaudos por hora (con representación gráfica). Reportes de recaudos diarios (con representación gráfica).

SICE	Smart Mobility	España	Tráfico total (con representación gráfica). Estadísticas (con representación gráfica). Reportes auditoria procesos servidores Linux (mensajes). (Convial, s.f.) Sistema BackOffice: Recogida de la información registrada por los equipos de campo. Verificación de datos, validación y finalización de estos. Tarificación y facturación. Gestión de vehículos no autorizados. Generación de Informes. Suministro de un gran número de servicios a clientes (SMS, fax, página web, etc.). Interfaces con entidades externas (Otras concesionarias, Agencias de gestión de infractores/deudores, agencias de cobro, bancos, canales de contacto con clientes, etc.). Emisión y gestión de stock y ciclo de vida de diferentes medios de pago (TAGs, tarjetas de la concesionaria, etc.) (SICE, s.f.)
i+D3	3TOLL	España	Software Estación: Servidor del área de peaje y base de datos cliente de liquidación de área de peaje. Cliente de supervisión de área de peaje: Tráfico por categoría. Tabla de Tiempo de tráfico. Lista de incidentes. Estadísticas: Cliente de mantenimiento de carril: Seguimiento de una operación y configuración carril, Telecomandos. Búsqueda de los incidentes registrados. (i+D3, s.f.)
TECSIDEL	Tecsidel Toll +	España	Herramienta web: Monitorea todos los equipos con un acceso seguro para diferentes perfiles de usuario. Esto permite: Una visualización general y detallada de los subsistemas y las últimas transacciones registradas. La notificación de incidencias en tiempo real mediante la generación de alarmas. Verificar el modo de operación de los dispositivos, configurar parámetros, ejecutar operaciones remotas y modificar el estado de los equipos. El sistema dispone de varios módulos como: Sistema de Monitorización y Control (MCS), control de recaudación y finanzas, y validación de transacciones. (Tecsidel, s.f.)

Nota. Fuente: Elaboración propia a partir de información de páginas web de cada empresa o clientes.

Este proyecto está orientado a mejorar el sistema existente de la Web Vial.

2.1.2. Impacto

Se establece que con el proceso a aplicar de arquitectura de software a la aplicación Web Vial, conlleve al cumplimiento de los objetivos estratégicos de la empresa dadas la visión y misión de la compañía, adicionalmente de llevar a una evolución del producto.

De acuerdo con lo anterior, la definición de una arquitectura y establecimiento claro de procesos se reflejará en las siguientes áreas que tiene que ver con la producción del software:

- *Área de Desarrollo*: permitirá que el equipo pueda realizar cambios de una forma definida y organizada, beneficiando y orientando tanto a miembros nuevos como antiguos.
- *Área de Pruebas:* al definir que las funciones de la aplicación se organicen por módulos, permitirá que los conjuntos de pruebas ejecutadas sean puntuales y dirigidas a las funcionalidades correspondientes a cada módulo y no siendo necesaria una prueba de regresión total, cada vez que se entregue una funcionalidad por parte del área de desarrollo.
- *Área de soporte:* Instalaciones y soporte puntual sin afectaciones generales del sistema.
- *Concesiones (Clientes):* La modularidad ofrecida por la plataforma beneficiará a los clientes en dos aspectos principales. El primero, aislará el desarrollo de soluciones a sus nuevas necesidades independiente a las funcionalidades actuales. El segundo, se garantiza el mantenimiento por actualizaciones de módulos sin necesidad de la afectación general del sistema y aplicar soluciones definitivas puntuales. Al lograr la

implantación estable de los módulos se logrará incrementar la confianza en el producto.

2.1.3. Componente de innovación

Mediante la modularizarían de la aplicación, actualización de las herramientas de desarrollo, y establecimiento de una arquitectura en la cual no se dependa de una tecnología exclusiva para el desarrollo, se buscará que la aplicación evolucione con calidad orientada a la innovación de funcionalidades de forma efectiva y eficiente cumpliendo y superando las expectativas de las concesiones que son los clientes del producto de software.

2.2. Marco teórico

Los siguientes conceptos son la base para el planteamiento y generación de la solución a los problemas identificados en los apartados anteriores de este documento, buscando que cada uno de ellos establezca un aporte importante en las diferentes fases que seguirá el proyecto.

2.2.1. Ingeniería de Software

Uno de los principales problemas observados en la construcción de la actual aplicación, es la falta de organización durante todo el ciclo de vida del software, es por ello que el primer concepto a tener en cuenta es el de Ingeniería de Software, que apoya todo el proceso y se define como "… una disciplina de la ingeniería que permite comprender todos los aspectos de la producción de software, desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de este después de que se utiliza.” (Somerville, 2005).

El SWEBOK es la guía de conocimiento que contiene las recomendaciones en los procesos de software, liderado por la IEEE Computer Society, establece que la ingeniería de Software se encuentra organizada en 15 áreas, cada una de estas áreas esta descrita por una visión general de su ámbito y como se relaciona con las demás áreas del conocimiento, consta de sub-áreas y subtemas que ofrecen un marco amplio de conocimiento de los tópicos enmarcados dentro de cada una para describir a gran nivel de detalle cómo funciona cada parte de la ingeniería de software.

Las 15 áreas de conocimiento son:

- Requisitos de software.
- Diseño de software.
- Construcción de software.
- Pruebas de software.
- Mantenimiento de software.
- Gestión de la configuración.
- Gestión de la ingeniería de software.
- Proceso de la ingeniería de software.
- Herramientas y métodos de la Ingeniería de Software.
- Calidad del software.
- Práctica profesional de la Ingeniería de Software.
- Economía de la Ingeniería de Software.
- Fundamentos de Computación.
- Fundamentos matemáticos.

- Fundamento de la Ingeniería.

Según lo anterior, en este proyecto se busca enfocar en los lineamientos de las áreas de requisitos, diseño, construcción, pruebas, mantenimiento y calidad, dejando la base para que pueda ser extendido y evaluado en las demás áreas posteriormente.

2.2.2. Scrumban

La gestión de proyectos se debe realizar en proyectos de cualquier tema y tamaño si se pretende llevar a cabo toda su ejecución a buen término. Las metodologías clásicas versus las agiles han tomado gran relevancia gracias a que permiten la organización del trabajo de una forma más flexible.

En el proceso realizado con el actual software se sigue una metodología tradicional en Cascada, lo cual permite organización de cierta forma, pero una de las desventajas es que el avance en sus diferentes etapas es dependiente una de la otra, y al aplicarla se ha convertido en un lastre, adicionalmente que en muchas ocasiones se ha detectado el cambio de los requerimientos sobre el avance, bien sea por la falta de entendimiento del requerimiento o por las variaciones de mismas de los procesos.

Realizando la consulta de las diferentes metodologías ágiles, la metodología Scrumban permite la combinación de dos metodologías Scrum y Kanban, también puede ser adaptada posteriormente al mantenimiento de la plataforma. Por un lado, Scrum permite definir procesos y busca la entrega continua de valor al cliente, teniendo en cuenta el concepto de generar productos entregables; por otro lado, Kanban permite la gestión de las tareas del proyecto trabajando en pro de la eficiencia y organización del proceso y mejora continua que puede ser otro factor aplicable a posteriores etapas de la plataforma.

2.2.3. Historias de Usuario

Es una técnica utilizada en el desarrollo de software, consiste en la redacción de un requerimiento en unas pocas frases, adicional a esto se utiliza el lenguaje común del usuario.

Las historias de usuario son ideales para desarrollos con metodologías agiles, definen las especificaciones del proyecto, posteriormente y debido a la prioridad asignada a cada una, se acomodan dentro de las diferentes iteraciones y se realizan los estimados en tiempos para la realización de cada historia, este proceso lo realizan los desarrolladores (Cohn, 2009).

Características de una historia de usuario:

Independiente: Pueden existir historias con tengan dependencias con otras, es necesario separar de manera que resulten lo más independiente posible, ya que esta independencia es clave a la hora de la etapa del diseño.

Negociable: Las pruebas de validación son la manera en que los clientes pueden verificar el alcance de la historia de usuario.

Valoradas por clientes o usuarios: Las historias de usuarios son importantes para clientes y usuarios, deben reflejar lo que ellos esperan, no lo que el desarrollador quiera hacer.

Estimables: Cuando se estiman los tiempos para cada historia de usuario es más fácil definir cuanto tiempo se tardará un proyecto.

Pequeñas: Deben ser pequeñas por la naturaleza iterativa de los proyectos donde se utilizan.

2.2.4. Arquitectura de Software

Ahora bien, si hay una definición de que tiene que hacer el sistema mediante los requerimientos funcionales, también se necesita definir cómo y que limitaciones puede tener el sistema lo cual está dado por los requerimientos no funcionales en lo cual se centra la arquitectura de software.

Entre las variadas definiciones se encuentra la siguiente: “La Arquitectura de Software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones.”(Clements, 1996)

2.2.5. Microservicios

Al realizar la evaluación de los problemas presentados en el actual proceso, otro inconveniente que se establece es la dificultad que está teniendo para crecer e innovar. Una de las causas es que la arquitectura de capas que trata de seguir la actual plataforma se ha convertido en el punto débil, debido a su crecimiento en complejidad por la cantidad de funcionalidades ofrecidas y que se han vuelto dependientes. Este problema ha sido atacado de cierta forma, al hacer desarrollos de aplicaciones de escritorio, y logra a nivel de funcionalidad el objetivo a nivel funcional se ha logrado, pero se ha llegado a la proliferación de estas aplicaciones de escritorio con el inconveniente de que no se encuentran articuladas y cada una debe gestionar el proceso de autenticación y autorización de usuarios de forma

independiente. Otro inconveniente con este modelo el acceso de las aplicaciones, están limitadas a ser accedidas en equipos exclusivos.

Luego de obtener las condiciones de los párrafos anteriores, y como objetivo principal que es lograr un sistema escalable, una de las opciones que mejor permiten lograr ese objetivo, es aplicar una arquitectura orientada a los microservicios. No existe una definición unificada sobre lo que son los microservicios y se utilizará la siguiente:

> Es un enfoque para desarrollar una sola aplicación como un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y comunicándose con mecanismos ligeros, a menudo una API de recursos HTTP. Estos servicios se basan en capacidades comerciales y se pueden implementar de forma independiente mediante maquinaria de implementación totalmente automatizada. (Fowler & Lewis, 2014).

Al realizar el desarrollo con esta arquitectura permite el objetivo de segregar las funcionalidades de acuerdo con las necesidades del negocio. Si bien, también tiene desventajas como el aumento de la complejidad, se busca que al aplicar patrones se reduzcan los inconvenientes y finalmente estas terminen superando las actuales desventajas.

2.2.6. Microfrontends

Luego de establecer una separación de responsabilidades en la parte de la lógica de la aplicación permitida mediante los microservicios en el backend, viene otra parte fundamental del servicio que son las interfaces de usuario, pues si bien la lógica ya se encuentra funcional y utilizable para cualquier proceso, la finalidad del proyecto no solo es interactuar con

servicios externos, si no que existen los usuarios del peaje que son los que hacen finalmente operación y a quienes se debe dirigir los esfuerzos para la utilización del sistema.

En la búsqueda de información de microservicios muy poco se toca el tema de la interacción de los usuarios finales con respecto a la interfaz gráfica, entendiendo que la finalidad es separar la lógica de las aplicaciones. Se observa que en general, la interfaz de usuario de las aplicaciones se trata como un solo componente que accede a varios microservicios, este concepto cae en el mimo problema de un monolito enfocado en la interfaz visual.

Como opción de solución a este inconveniente aparece en el radar el concepto de microfrontends, definido como "Un estilo arquitectónico en el que las aplicaciones frontend se pueden entregar de forma independiente componentes de un todo mayor" (Jackson, 2019). Con este concepto se busca obtener la mayor modularidad posible, adicionalmente de apoyar a los equipos de desarrollo a trabajar de forma independiente.

3. Metodología

La metodología de proyectos a seguir en el desarrollo del proyecto es Scrumban. El objetivo de utilizar esta metodología es organizar, visualizar y establecer los flujos de las actividades del proyecto, permitiendo avances mediante la liberación de entregables como documentación, prototipos y o versiones de los diferentes módulos planteados y como factor adicional se tiene la limitación de recursos humanos trabajando en el proyecto, siendo esta aplicable a equipos pequeños y proyectos en los que se requiere agilidad y posterior mente se puede adaptar para dar mantenimiento y refinar las funcionalidades de la plataforma con el fin de realizar una mejora continua.

La implementación se realizó con ayuda de la herramienta Planner de Microsoft Office 365 a la cual se tienen acceso en los recursos brindados por la compañía, mediante la creación de un plan que representa el tablero del proyecto para la organización de las tareas en los siguientes sprints:

Tabla 2

Sprints de Planeación Scrumban

Sprint	Descripción	Artefactos Entregables	Duración (*)
WVP-Backlog	Creación de las diferentes tareas que se establezcan en cada iteración.	N/A	N/A
WVP-Sprint1	Sprint 1, el cual contiene las tareas de la fase inicial de requerimientos.	Casos de Uso, Historias de usuario	4 semanas
WVP-Sprint2	Sprint 2, el cual contiene las tareas de la fase inicial de diseño.	Software Document Architecture	2 semanas
WVP-Sprint3	Sprint 3, el cual contiene las tareas del módulo de administración de usuarios.	Ejecutable módulo administración de usuarios	4 semanas
WVP-Sprint4	Sprint 4, el cual contiene las tareas del módulo navegación.	Ejecutable módulo navegación	4 semanas

Sprint	Descripción	Artefactos Entregables	Duración (*)
WVP-Sprint5	Sprint 5, el cual contiene las tareas del módulo principal encargado de articular los diferentes.	Ejecutable modulo principal integrando módulo de navegación y administración de usuarios.	4 semanas
WVP-Sprint6	Sprint 6, el cual contiene las tareas del módulo de configuración de estación.	Ejecutable módulo de configuración de estación.	4 semanas
WVP-Sprint7	Contiene las tareas de validación de los aspectos de calidad.	Evidencias chequeos análisis estático de código, cobertura, pruebas unitarias y de integración	1 semana
WVP-Sprint8	Contiene las tareas de validación de los aspectos de seguridad.	Evidencias chequeos OWASP	1 semana

Nota. Fuente: elaboración propia.

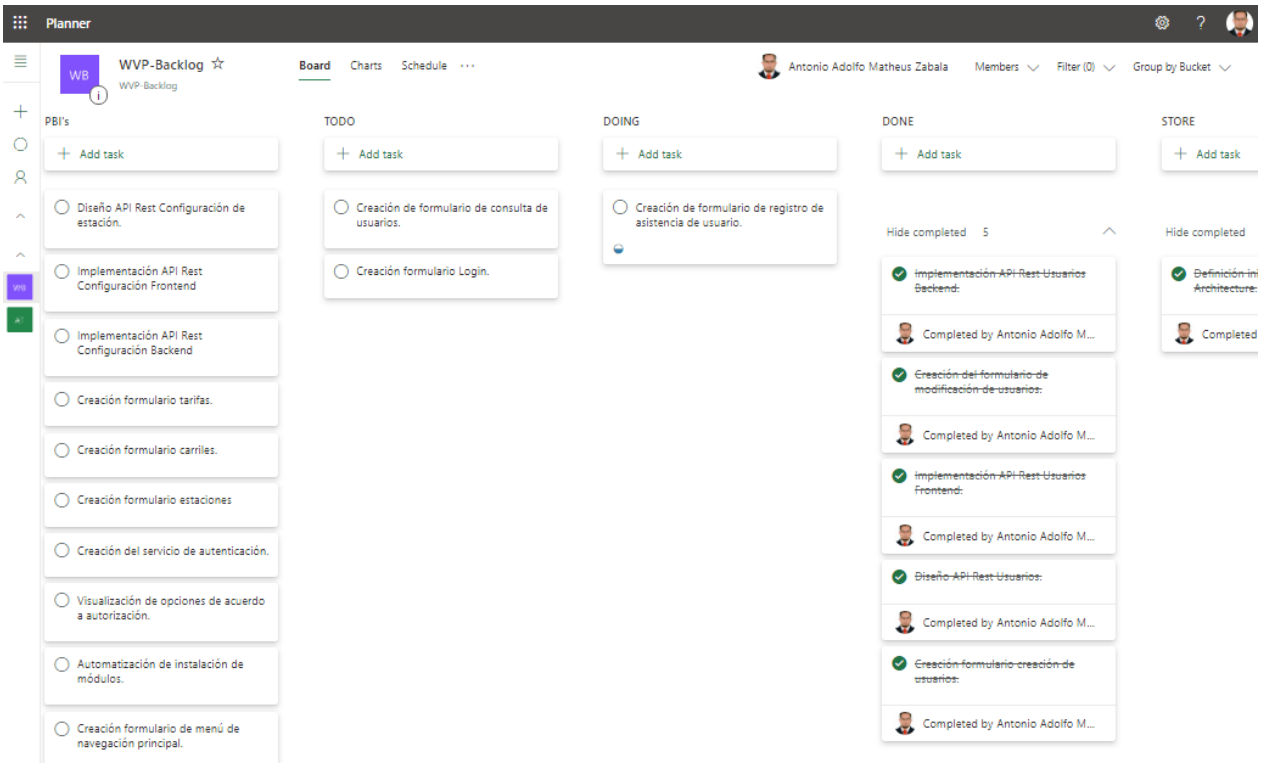
La estructura del tablero contiene 5 secciones:

- PBI's: Product Backlog Item's en la cual se van creando las tareas que se consideren necesarias o según se vayan refinando en las iteraciones.
- TODO: En esta sección se colocan las tareas correspondientes a una iteración al iniciar el correspondiente sprint y que se deben cumplir para dar por terminado dicho sprint.
- DOING: En esta sección se colocan las tareas que se están realizando diariamente correspondientes al sprint en curso.
- DONE: En esta sección se colocan las tareas que se dan por finalizada su ejecución en el sprint en ejecución.
- STORE: Cuando un sprint se pueda dar por finalizado completamente se pasan las tareas finalizadas de ese sprint por un periodo de dos meses a esta sección antes de ser eliminada la tarea definitivamente del plan de trabajo, con el fin de tenerla para una posible retroalimentación o el simple registro de actividades del equipo.

En la siguiente imagen se puede observar un instante de ejecución para el sprint número 3 correspondiente a la administración de usuarios:

Figura1

Tablero de Tareas Scrumban con Planner



Nota. Fuente: Elaboración propia.

En el anexo 0 se presenta un planteamiento básico para los requerimientos de seguridad del desarrollo y que ayudaron a plantear algunos requisitos de la implementación, basado en la metodología MSDL (Microsoft Security Development Lifecycle), la cual "introduce consideraciones de seguridad y privacidad en todas las fases del proceso de desarrollo, lo que ayuda a los desarrolladores a crear software altamente seguro, abordar los requisitos de cumplimiento de seguridad y reducir los costos de desarrollo."(Microsoft, s. f.).

Adicionalmente en el anexo 0 se establecen los aspectos relevantes con respecto a la estrategia de gobierno, que por ser un proyecto orientado a una solución empresarial vale la pena presentarlo.

4. Proceso de Software

4.1. Requerimientos funcionales

Web Vial Plus gestiona varios conjuntos de funcionalidades para la administración de operación y recaudo de Peajes como un sistema de módulos o plugins que son articulados a través de una aplicación central, lo cual permite instalar, actualizar o eliminar módulos de funcionalidades independientes sin la afectación general de la plataforma. Cada módulo tiene interfaces de usuario con acciones establecidas mediante menús y vistas con accesos asignables a perfiles de usuario en la opción de permisos en el módulo de gestión de acuerdo con las necesidades del cliente, estableciendo las restricciones de acceso a las operaciones para los diferentes cargos de colaboración en el proceso.

Se establecen los siguientes módulos:

- Aplicación central.

Este módulo se encarga de la instalación, actualización, registro, eliminación y consulta del estado y versiones, de los módulos funcionales que cuenten con la interfaz de integración soportada por la plataforma.

- Gestión de Usuarios.

Este módulo agrupa las funcionalidades de gestión para la operación de un peaje entre los que se incluyen:

Creación y actualización e inactivación de usuarios administrativos y operativos.

Enrolamiento e inactivación de usuarios, el cual es el proceso de asignación de grupo estación y tarjeta de usuario.

Creación, actualización e inactivación de perfiles.

Asignación, modificación y eliminación de los permisos a los perfiles de acuerdo con cada una de las opciones disponibles en los distintos módulos instalados y activados en la aplicación.

Registro de la asistencia y dineros personales de los usuarios en las estaciones de peaje.

- Configuración de estación

Actualización de la información básica de la estación de peaje.

Creación y actualización de tarifas de peaje.

Creación y actualización de los horarios de turnos operativos del peaje.

Creación y actualización de los carriles para operación de la estación de peaje.

Consideraciones generales.

Todos los usuarios registrados en el sistema con estado activo pueden registrar la asistencia y dineros personales al entrar o salir de una estación de peaje para llevar el registro y control de asistencia de una estación.

La administración de usuarios debe estar a cargo de grupo de usuario de responsabilidad alta, tal y como lo es un jefe o supervisor de peaje.

La administración de grupos de usuario y permisos de las opciones de la aplicación, administración de tarifas y administración de turno operativos, debe estar a cargo de un grupo de usuario superior como lo es el Director de Operaciones.

Los permisos sobre la consulta de las opciones de reportes son asignados por el gerente de operación de acuerdo con las necesidades de la operación de la estación de peaje.

En la siguiente tabla se recopilan las historias de usuario identificadas de los requisitos funcionales, siguiendo la forma de descripción *Como <usuario> quiero <objetivo> para <beneficio>*

Tabla 3

Historias de Usuario Identificadas

ID	Usuario	Objetivo	Beneficio
HU01	En estado Activo.	Ingresar las credenciales de autenticación en el sistema y que sean verificadas.	Obtener el acceso a las opciones de la plataforma habilitadas según el perfil.
HU02	Jefe/Supervisor de Peaje	Se presente un formulario para la creación de usuarios con la siguiente información: *Requerida*: número de identificación, nombres, apellidos, fotografía, aceptación de la política de tratamiento de datos personales. *Opcional*: Dirección, Teléfono, Correo electrónico, Nombre persona para contacto de emergencia, Teléfono persona contacto de emergencia.	Crear nuevos usuarios para la operación y administración de las estaciones de peaje.
HU03	Director de Operaciones	Se presente un formulario para asignar el perfil y la estación a usuarios creados en la plataforma	Activar usuarios estableciendo las opciones requeridas para el cumplimiento de sus funciones en las estaciones de peaje.
HU04	Jefe de Peaje	Se presente un formulario para la asignación de tarjeta de ingreso y	Permitir la definición de credenciales de autenticación

ID	Usuario	Objetivo	Beneficio
		restablecimiento de contraseñas de usuarios activos	mediante Tarjeta en las aplicaciones de carril.
HU05	Director de Operaciones/Jefe de Peaje	Se permita obtener un listado de usuarios con las siguientes opciones de criterios de búsqueda: número de identificación, nombre, apellido, estación, perfil, estado.	Ver los detalles de cada usuario y/o realizar modificaciones de la información de un usuario seleccionado.
HU06	Director de Operaciones	Presentar un formulario que permita la creación de nuevos perfiles para la operación del sistema, con la siguiente información: - Nombre - Estado	Permitir restringir las opciones de plataforma que un conjunto de usuarios activos pueda realizar al asignar dicho perfil.
HU07	Director de Operaciones	Se permita obtener el listado de perfiles registrados con las siguientes opciones de criterios de búsqueda: nombre, estado.	Ver los detalles de cada perfil y realizar modificaciones de un perfil seleccionado.
HU08	Director de Operaciones	Presentar un formulario con la información detallada de un perfil seleccionado y se permita la modificación del nombre y/o estado y la asignación o revocación de permisos de acceso a opciones de la plataforma.	Gestionar el acceso de los usuarios a las opciones del sistema y personalizar el perfil.
HU09	En estado Activo	Formulario para registrar la asistencia a una estación de peaje ingresando el valor de dinero cuando se ingresa a la estación y la notificación del saldo cuando se retira de la estación.	Llevar el libro de control de asistencia y dineros personales de la estación de peaje.
HU10	Director de Operaciones	Formulario para la actualización de la información de estación como lo es, nombre de los sentidos, nombre de la estación, nombre de concesión, NIT, Contrato.	Permitir la personalización de la información de estación de peaje.
HU11	Director de Operaciones	Se permita obtener el listado de turnos operativos con la opción de modificar o crear nuevos turnos validando la continuidad de horarios.	Permitir la personalización de operación de la estación de peaje.
HU12	Director de Operaciones	Se permita obtener la lista de tarifas registradas en la estación y permita la creación de estas llevando el histórico.	Permitir la actualización de las tarifas a que haya lugar por resoluciones expedidas por las autoridades regulatorias.
HU13	Director de Operaciones	Se permita obtener la lista de los carriles que se encuentran registrados en la estación	Permitir la expansión de operación de la estación

ID	Usuario	Objetivo	Beneficio
		con la opción de modificar carriles existentes o la creación de nuevos.	mediante la apertura de carriles.
HU14	Director de Operaciones	Al obtener el detalle de un usuario, exista la opción de inactivarlo cambiando el estado, y automáticamente quitando las asignaciones que tenga de perfil, estación y tarjeta de ingreso.	Permitir el retiro de personal de operación de las estaciones de forma segura.
HU15	Administrador de Plataforma	Realizar la instalación de módulos que sean invocados desde una aplicación central y que realice el correspondiente registro de opciones con que cuenta el módulo.	Permitir extender las funcionalidades de la plataforma sin necesidad de tener que actualizar toda la plataforma.
HU16	Administrador de Plataforma	Se permita actualizar los módulos instalados.	Permitir mantener de forma modular la plataforma sin necesidad de tener que actualizar toda la plataforma.
HU17	En estado Activo	Visualizar el menú con las opciones de la plataforma habilitadas según el perfil asignado.	Permitir realizar las labores implicadas en el desarrollo de las funciones asignadas en la estación de peaje.

Nota. Fuente: elaboración propia.

4.2. Requerimientos no funcionales

A continuación, se establecen los requisitos no funcionales, es decir, aquellas características de calidad y restricciones que debe garantizar la aplicación. Se establecen los más relevantes y se consideran transversales a toda la aplicación.

Tabla 4

Identificación Requerimientos no Funcionales

Aspecto	Requerimiento
Rendimiento	- El sistema permitirá conexiones concurrentes a la aplicación web, mínimo 20. - El tiempo para carga de la aplicación cuando se acceda a la página principal no será mayor a 5 segundos, y el tiempo de navegabilidad entre pantallas será de máximo 5 segundos. - Se deberá optimizar las imágenes y cantidad de datos que viajan entre el cliente y el servidor.

Aspecto	Requerimiento
Robustez	- El sistema contará con un sistema de manejo de errores frente a eventos no planificados, para lo cual se utilizará Log4J para registrar los errores. - Para la gestión de datos y para asegurar la correcta actualización de los datos se utilizará el framework Hibernate, que permite al programador abstraerse del manejo transaccional y centrarse pura y exclusivamente en las operaciones de lógica de negocios.
Seguridad	- El sistema contará con un módulo de seguridad, encargado de la autenticación de usuarios y autorización de las operaciones. La autenticación se realiza mediante contraseña, y tras de ser autenticado se genera un token con el método JWT, en el cual debe contener toda la información necesaria para la autorización de posteriores operaciones en el sistema, teniendo la capacidad de ser validado localmente por cada microservicio. - Los usuarios deberán estar registrados y autenticados. Solo un usuario autenticado podrá realizar las operaciones a las que tenga garantizado el acceso el grupo de usuario asignado. - Las contraseñas deberán cumplir un nivel de complejidad tener 8 caracteres como mínimo y usar mayúsculas y minúsculas. - Las contraseñas debe ser encriptadas antes de ser almacenadas, para eso se utilizará el algoritmo Bcrypt. - Los datos sensibles, tales como plantillas de huelles dactilares, deben ser encriptadas antes de ser almacenadas, para eso se utilizará el algoritmo 3DES. - Las conexiones de comunicación deben ser seguras utilizando HTTPS con soporte de certificados con protocolo TLS 1.2 o posterior. - Se debe tener presente la presentación de políticas de tratamiento de datos debido a la captura de datos sensibles para la debida autorización del personal registrado en la plataforma.
Escalabilidad	- La plataforma debe permitir modularidad con el fin de poder instalar o actualizar partes del sistema sin la necesidad de afectación total y a la vez permitir ser articuladas y gestionadas por un módulo central y ser capaces de gestionarse por un mismo sistema de autorización y autenticación. Los módulos deben tener funciones relacionadas con un objetivo claro y definido del sistema. - El diseño del sistema y su construcción deberá contemplar la división entre los datos y la lógica de la aplicación para optimizar la escalabilidad de la aplicación. - Se utilizará el framework Hibernate para mantener independiente el motor de base de datos y su ubicación. - Esta estructura permite no sólo escalabilidad y reusabilidad, sino también fácil detección de errores además constituye una buena práctica para el desarrollo de aplicaciones empresariales.
Diseño	- El sistema permitirá cambiar su estilo a través de hojas de estilo en cascada (CSS), que deberán personalizarse para cada componente, añadir nuevos estilos y extender componentes para la reutilización de funcionalidad. - Se utilizará las ventajas de la especificación de HTML5. - No se modificará el código de los frameworks para que la aplicación soporte actualizaciones a nuevas versiones. - La aplicación debe tener un diseño Responsive con el fin de poder garantizar la correcta visualización del contenido en distintas resoluciones de pantalla en dispositivos.

Aspecto	Requerimiento
Usabilidad	- Los mensajes de error deben ser concisos e informativos con un lenguaje natural para el usuario final y no deben contener información técnica o detalles de los procesos ejecutados. - La presentación de mensajes para validación deben ser concretos y visualizados debajo del campo validado en color rojo.

Nota. Fuente: elaboración propia.

4.3. Diseño y arquitectura

Mediante la identificación en la sección anterior de los requerimientos no funcionales del sistema, se establen los siguientes drivers de arquitectura para el diseño de la solución: rendimiento, robustez, seguridad, escalabilidad, diseño y usabilidad. Para realizar diseñar la solución se siguió un modelo de arquitectura basada en microservicios, logrando desacoplar las funciones mediante la definición de API´s Rest, y definiendo separación de responsabilidades.

Al realizar una separación de la lógica de la aplicación a través de microservicios, queda la incógnita de cómo lograr este mismo comportamiento en la parte de las interfaces visuales, pues no se quiere que la parte del frontend se convierta en una aplicación monolítica que conlleve a un problema similar que se tiene con la actual aplicación, si bien solo se afectaría la interfaz gráfica, el driver de escalabilidad requiere implementar nuevas funcionalidades como módulos independientes. Para lograr lo anterior se aplica el concepto de microfrontends, que actualmente está tomando importancia al aplicar los principios de los microservicios a la construcción de las interfaces de usuario, permitiendo desacoplar también la construcción de la interfaz de usuario.

Adicional a los atributos de calidad establecidos en la sección anterior, también se busca cumplir los siguientes atributos de negocio:

- Time To Market: poder realizar la entrega de funcionalidades a los clientes en tiempo de entre 2 a 4 semanas.
- Costo-Beneficio: Para este atributo se podrá medir en varios aspectos. Inicialmente reducir el soporte de aplicaciones al permitir disminuir las fallas en producción por ausencia de calidad de la aplicación. Segundo, permitir que los recursos dedicados al mantenimiento de la aplicación se puedan orientar a desarrollar nuevas soluciones innovadoras que permitan expandir el producto de software ofreciendo funcionalidades que creen valor para el cliente. Tercero, los dos aspectos anteriores impactaran en la recuperación de la confianza de cliente existentes que se pueda ver reflejado en renovación y creación nuevos contratos y ventas de la plataforma. Un cuarto punto tiene impacto directamente en el costo-beneficio del cliente y consiste en que se reduce el costo de la plataforma para el cliente gracias a la posibilidad de la eliminación de la necesidad de licencias de bases de datos Oracle, debido a que la actual plataforma tiene una dependencia directa y se requiere adicionar el valor de estas licencias al costo de la venta; con la posibilidad de uso de otros motores de bases de datos gratuitas, se obtiene el mismo beneficio con la reducción del costo, pero también con la posibilidad que si un cliente prefiere la base de datos Oracle se puede utilizar, claramente con el incremento del costo.
- Ciclo de vida de la solución: Al dividir la plataforma en módulos, cada uno de ellos puede ser tratado como un proyecto independiente siendo medibles de forma más precisa y no como en la actualidad donde los requerimientos hacen parte de un mismo producto lo cual dificulta la valoración total de la plataforma.
- Lanzamiento de producto: Nuevamente se menciona el modularidad que permite que el producto se pueda ofrecer de acuerdo con las necesidades del cliente, como en la

actualidad, pero con la diferencia que solo se entrega lo que realmente se adquirió por parte del cliente. En la actualidad cabe la posibilidad que en producción se activen funcionen que no están dentro de las condiciones iniciales del contrato de venta.

4.3.1. Diagrama de despliegue.

De acuerdo con las características de la infraestructura de los peajes colombianos, en el cual las estaciones en algunas ocasiones se encuentran aisladas (bien por seguridad o por limitaciones técnicas) sin conexión a Internet, se debe garantizar que la plataforma pueda ser operada en una red de área local (correspondiente al alcance de la estación), es requerido que se realice un despliegue en servidores locales de la estación de peaje.

En las estaciones de peaje se cuenta con la infraestructura para tener por lo menos un servidor de estación, el cual debe contar con unas características robustas para soportar la instalación del servidor de aplicaciones y el motor de base de datos sobre las cuales se va a soportar la plataforma. En esta fase de despliegue no se tiene en cuenta las características de respaldo y comunicaciones de toda la plataforma, se tiene en cuenta solo lo relacionado con la instalación y operación de la aplicación Web Vial Plus.

Las características mínimas requeridas del servidor son:

Tabla 5

Características Mínimas para Despliegue

Característica	Valor
Hardware	
Memoria RAM	4 GB

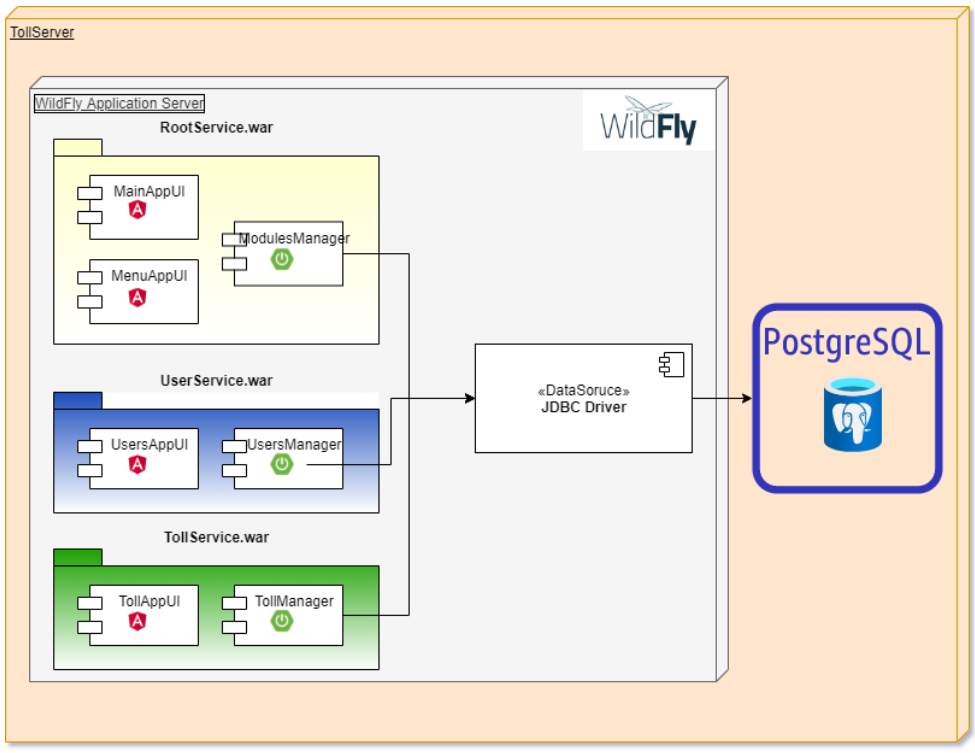
Almacenamiento libre	32 GB
Procesador	Intel Core i5, AMD Ryzen 5
	Software
Sistema Operativo	Multiplataforma
Java	JDK 8
Servidor de aplicaciones	WildFly Application Server
Base de datos	PostgreSQL 12 Alternativa: Oracle 18c Express Edition

Nota. Fuente: elaboración propia.

En el siguiente diagrama se presenta el despliegue de componentes:

Figura 2

Diagrama de Despliegue Aplicación Web Vial Plus



Nota. Fuente: elaboración propia.

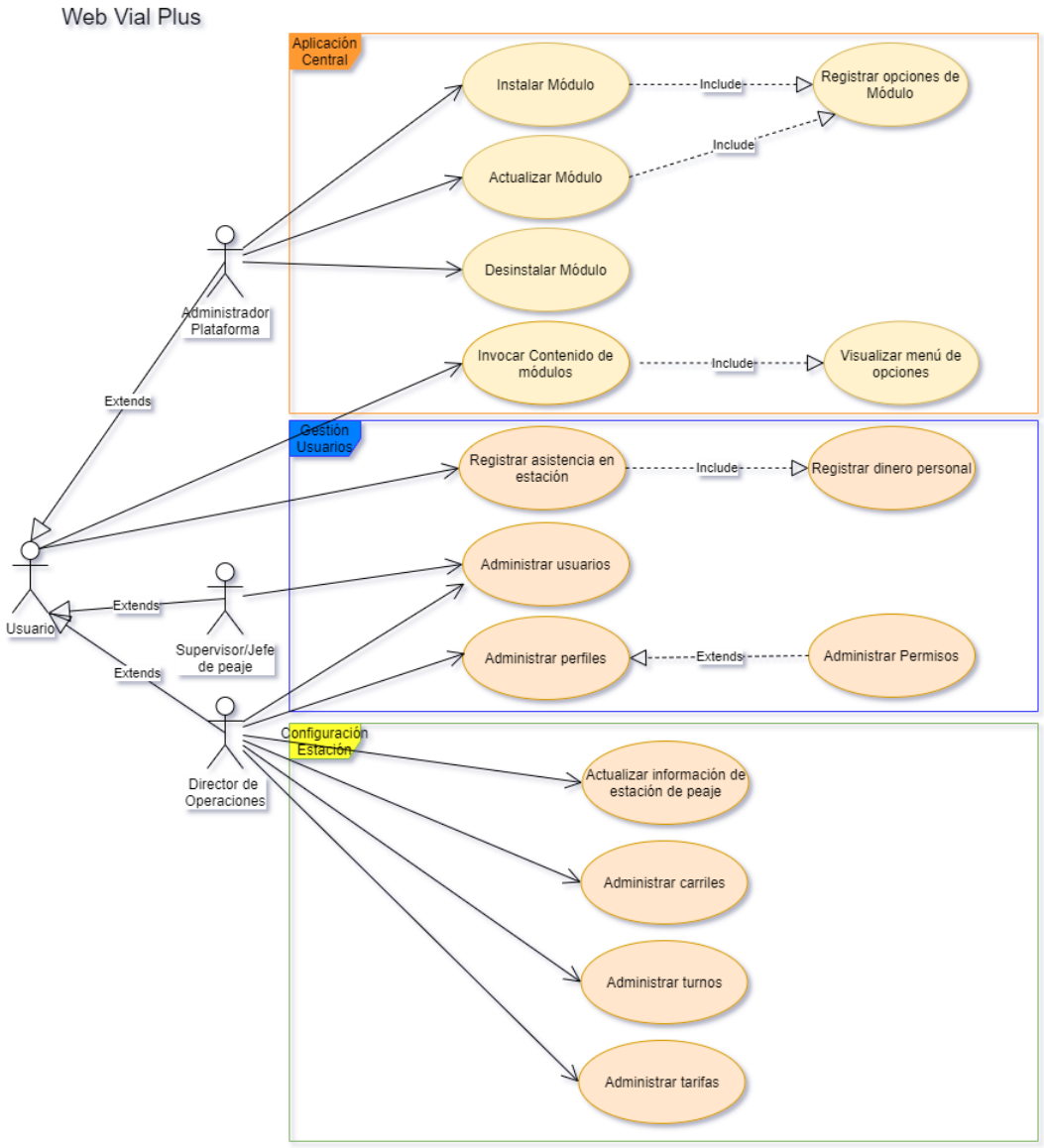
4.3.2. Caso de uso arquitecturalmente relevante.

En la siguiente imagen se observa el caso de uso que involucra las actividades que el sistema ofrece, junto con los actores que participan. Se realiza la división de los casos de uso por marcos que establecen los límites de cada uno de los módulos a desarrollar.

En el diagrama no se involucran los casos de uso que implícitamente se deben cumplir que corresponden a requisitos no funcionales como los casos de uso relacionados con la seguridad, es decir funciones de autenticación y autorización.

Figura 3

Casos de Uso Relevantes



Nota. Fuente: elaboración propia.

4.3.3. Diagrama de secuencia

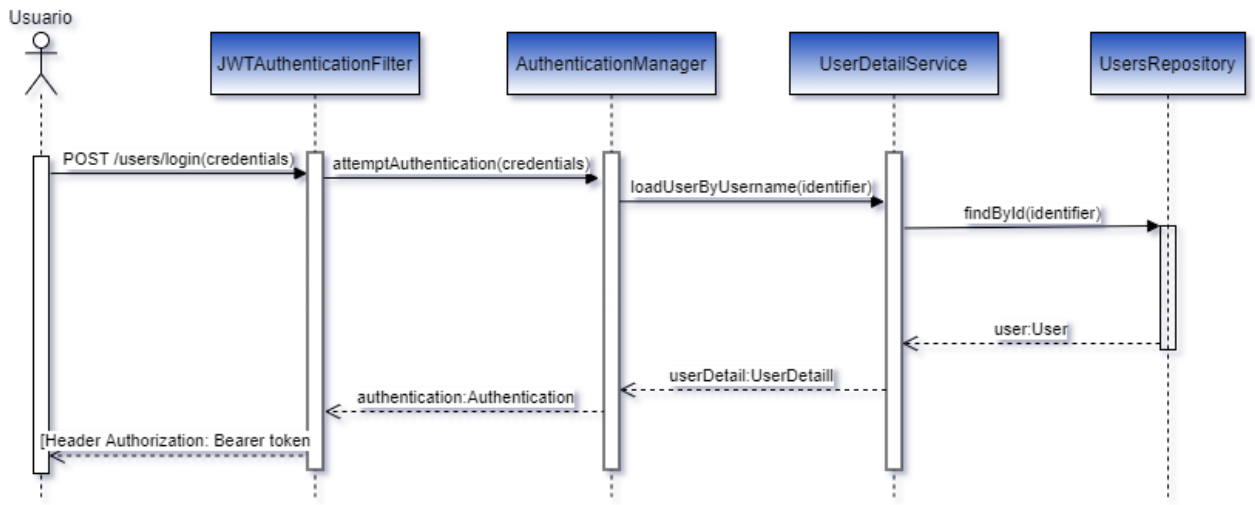
En la siguiente sección se presentan los diagramas de secuencia para los procesos relevantes del sistema.

La gestión de autenticación y autorización de los usuarios está basada en JSON Web Token que es "(JWT) es un estándar abierto (RFC 7519) que define una forma compacta y autónoma de transmitir información de forma segura entre las partes como un objeto JSON. Esta información se puede verificar y confiar porque está firmada digitalmente. Los JWT se pueden firmar usando un secreto (con el algoritmo HMAC) o un par de claves pública / privada usando RSA o ECDSA" (auth0.com, s. f.).

Por una parte, la autenticación de usuarios se realiza tanto para usuario con funcionalidades y operaciones con interfaz de usuario como también para usuarios que consumen las API. El token es generado si la autenticación es correcta y contiene la identificación del usuario y los privilegios asignados al perfil del usuario al que pertenece, adicional a información de vigencia de este. El proceso se presenta el siguiente diagrama de forma general; la funcionalidad es prestada por el módulo de usuarios:

Figura 4

Secuencia para Autenticación de Usuarios

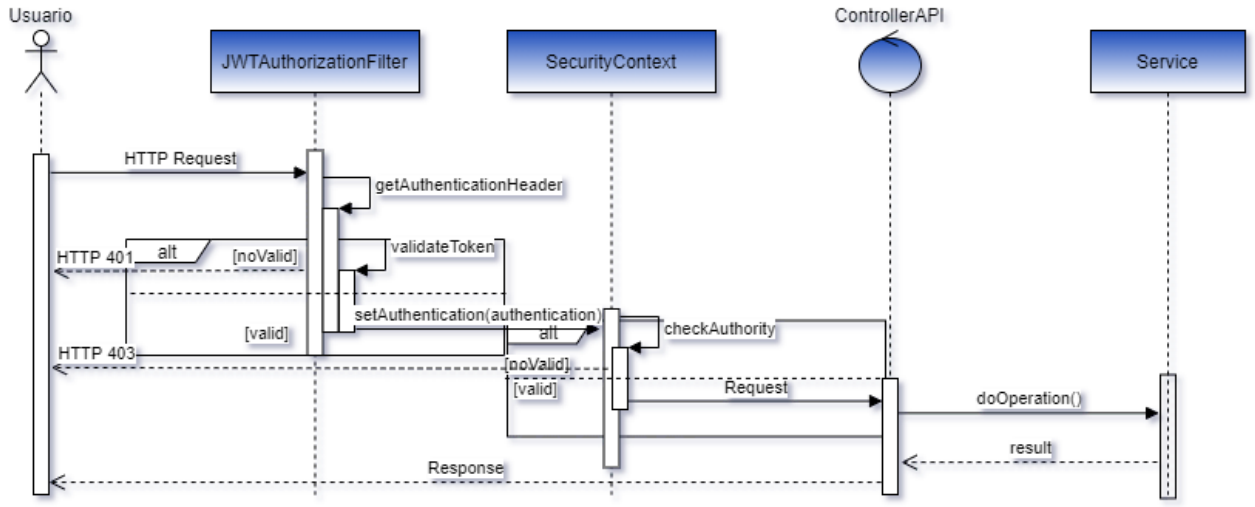


Nota. Fuente: elaboración propia.

Como siguiente proceso, se establece la autorización de los usuarios al acceder a recursos u operaciones de la plataforma. El token generado por el proceso anterior de autenticación debe ser enviado por el cliente en cada solicitud. Cada módulo debe poder verificar su validez localmente, y luego de ello extrae la información necesaria para identificación del usuario y validación de privilegios. En el siguiente diagrama se observa el este proceso con una petición genérica, la cual deben seguir todas las operaciones de los módulos implementados en el sistema:

Figura 5

Secuencia para Autorización de acceso a Recursos

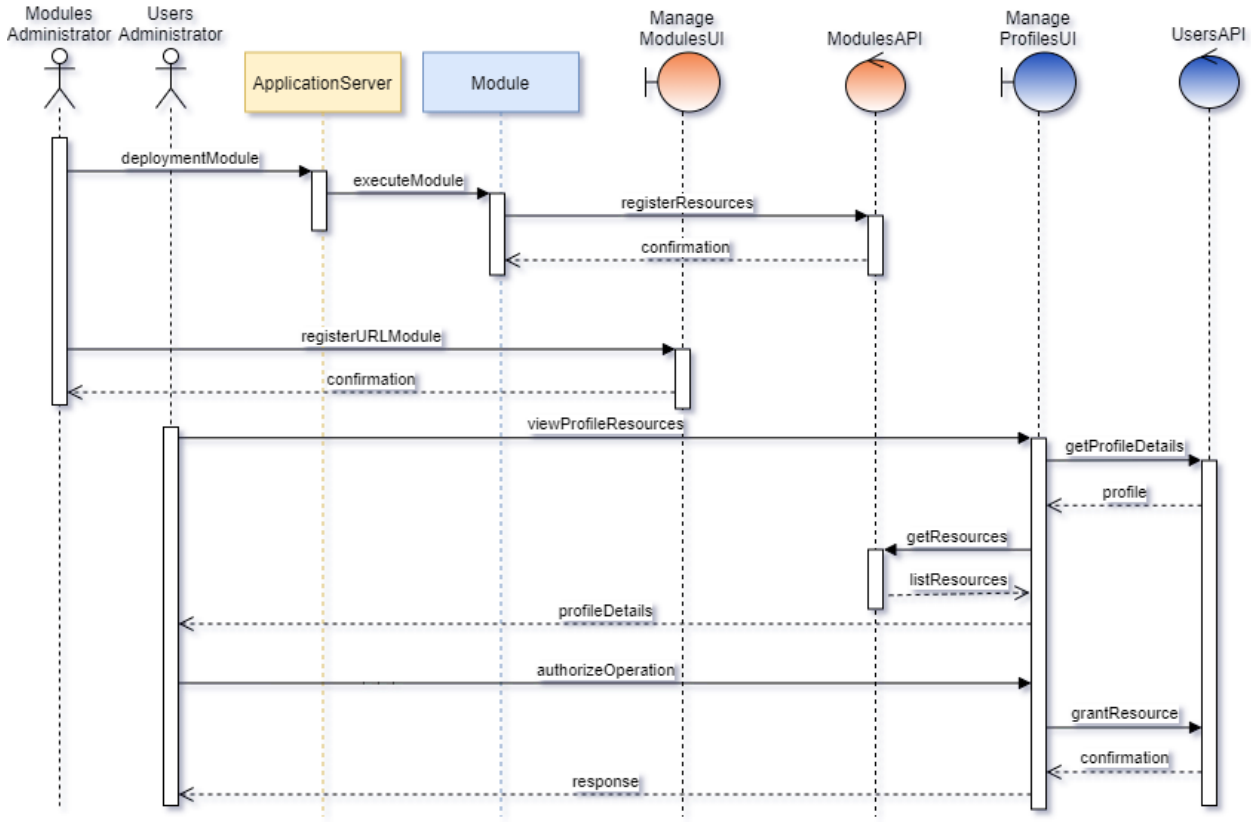


Nota. Fuente: elaboración propia

A continuación, se presenta el proceso de instalación de módulos, mediante el cual es posible además de adicionar nuevos módulos a la plataforma, gestionar los privilegios de acceso a las funcionalidades pertenecientes a ese nuevo módulo:

Figura 6

Secuencia de Instalación de Módulos



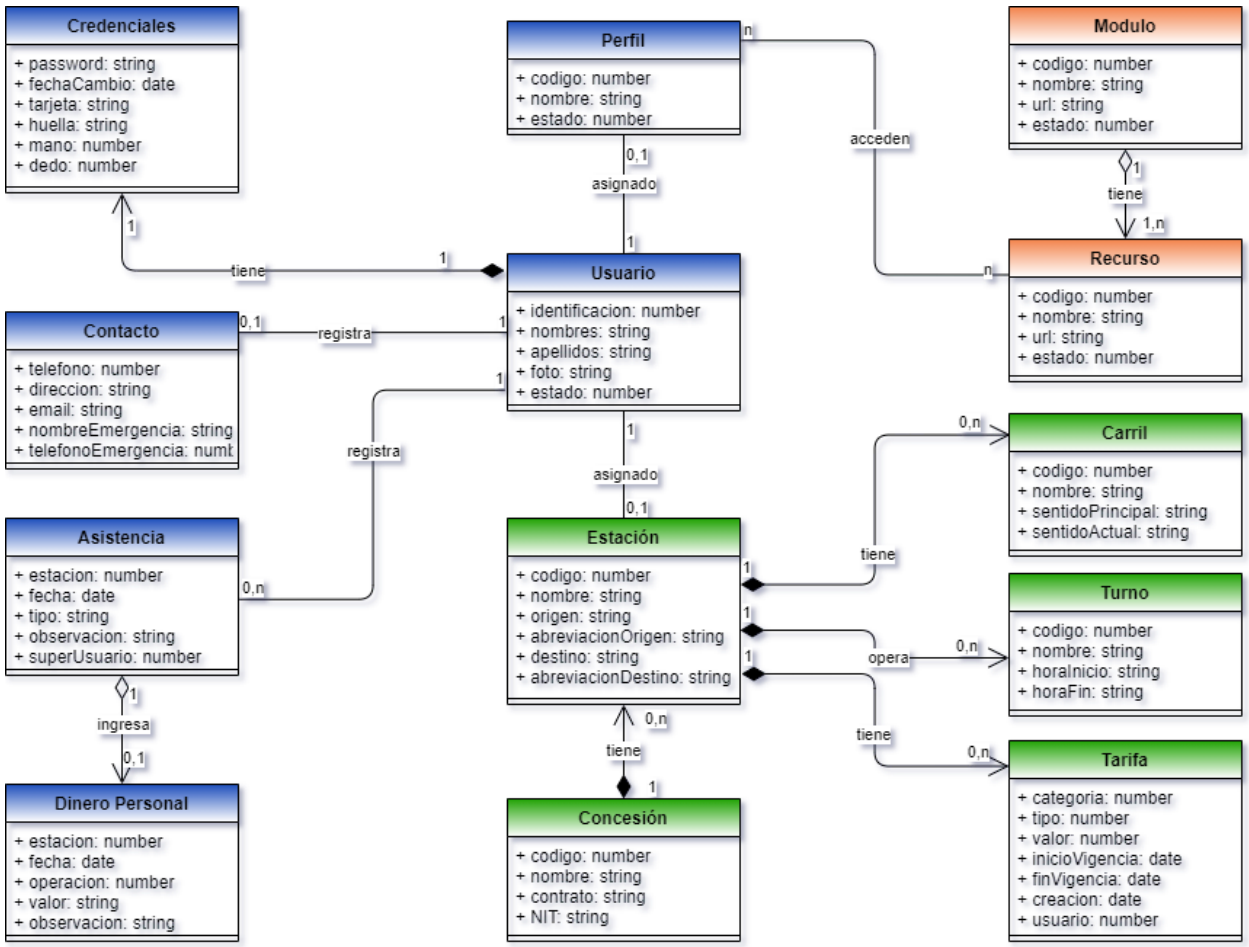
Nota. Fuente: elaboración propia

4.3.4. Diagrama de clases

En el siguiente diagrama se presentan los objetos representativos del sistema, con el fin de modelar de forma estática los objetos que intervienen en la lógica de los procesos de negocio:

Figura 7

Diagrama de Clases



Nota. Fuente: elaboración propia.

4.3.5. Arquitectura de alto nivel

La presente solución pertenece al conjunto de servicios establecidos para cumplir las funcionalidades de la aplicación Web Vial Plus, siguiendo un patrón de arquitectura basada en microservicios.

El sistema este compuesto de los siguientes módulos:

Modulo Home: Contiene la interfaz de usuario principal encargada de invocar y realizar las tareas de montaje y desmontaje de las interfaces de usuario de los módulos registrados en el sistema según las elecciones que el usuario realice en el menú de la aplicación; dicho menú se arma dinámicamente acorde a los permisos de acceso asignados al perfil de usuario que ingresa. También presenta la ventana para autenticación de los usuarios.

La siguiente función del este módulo es exponer el API de Módulos con el cual se realiza la instalación, actualización y desinstalación de módulos en la plataforma, adicional a prestar los servicios de descubrimiento, registro y monitoreo de los módulos, sirviendo como visualizador del estado de estos.

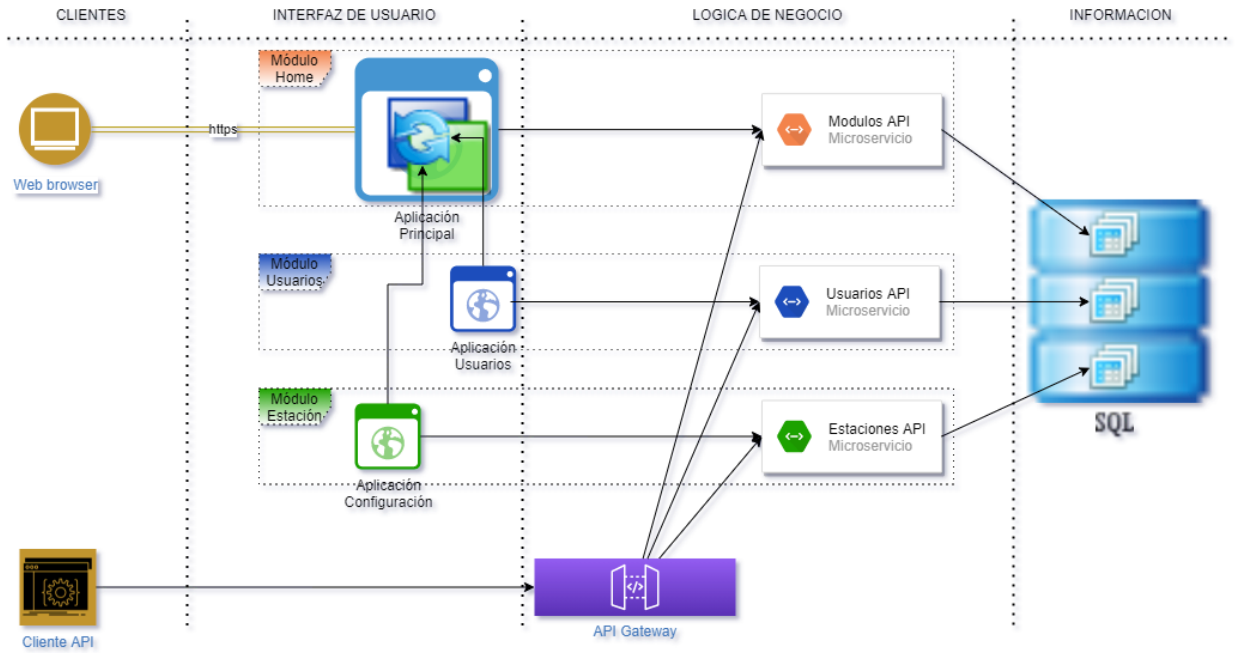
Modulo Usuarios: Contiene las interfaces de usuario y expone los recursos de la API Rest, relacionadas con la administración de usuarios de la plataforma, incluyendo la administración de perfiles y permisos de acceso.

Modulo Estación: Contiene las interfaces de usuario y expone los recursos de la API Rest relacionadas con la configuración de estaciones de peaje, teniendo en cuenta desde información general correspondiente a la Concesionaria Vial y luego aspectos más específicos como lo son los carriles, categorías vehiculares y tarifas de peaje.

En la siguiente imagen se presenta la estructura de alto nivel de la aplicación, en la cual se permite visualizar las interacciones de los diferentes componentes pertenecientes a la plataforma y como los usuarios finales acceden a los servicios prestados:

Figura 8

Diagrama de Arquitectura de Alto Nivel



Nota. Fuente: elaboración propia.

La estructura anterior permite realizar la instalación de nuevos módulos, los cuales deben seguir las siguientes condiciones para la integración a la plataforma:

- Las interfaces de usuario deben estar desarrolladas con frameworks basadas en Javascript, debido a que la función de gestión de aplicaciones de la aplicación principal se encuentra basada en funciones de este lenguaje.
- Un módulo estándar se compone de la exposición de un microservicio y contener las interfaces visuales para las funcionalidades que vaya a prestar el módulo. Sin embargo, es posible que un módulo se componga solo del microservicio (prestando solamente funciones de lógica de negocio), pero, no es posible instalar un módulo que solo contenga interfaces de usuario.

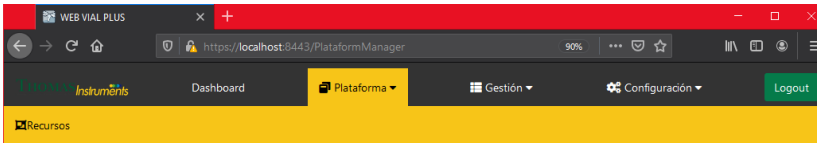
4.4. Construcción

En la siguiente sección se visualizan aspectos relevantes de la construcción de la aplicación.

Se visualizan imágenes del software:

Figura 9

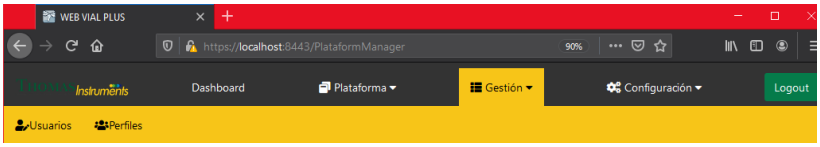
Menú de Plataforma



Nota. Fuente: elaboración propia.

Figura 10

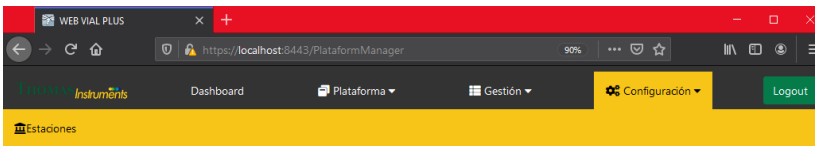
Menú de Gestión



Nota. Fuente: elaboración propia.

Figura 11

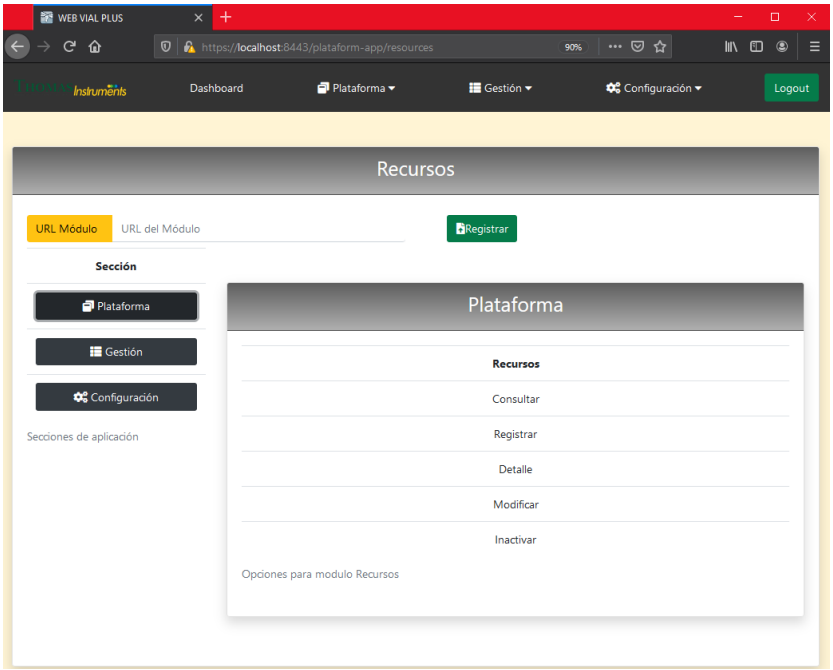
Menú de Configuración



Nota. Fuente: elaboración propia.

Figura 12

Módulo Administración de Recursos de Módulos



Nota. Fuente: elaboración propia.

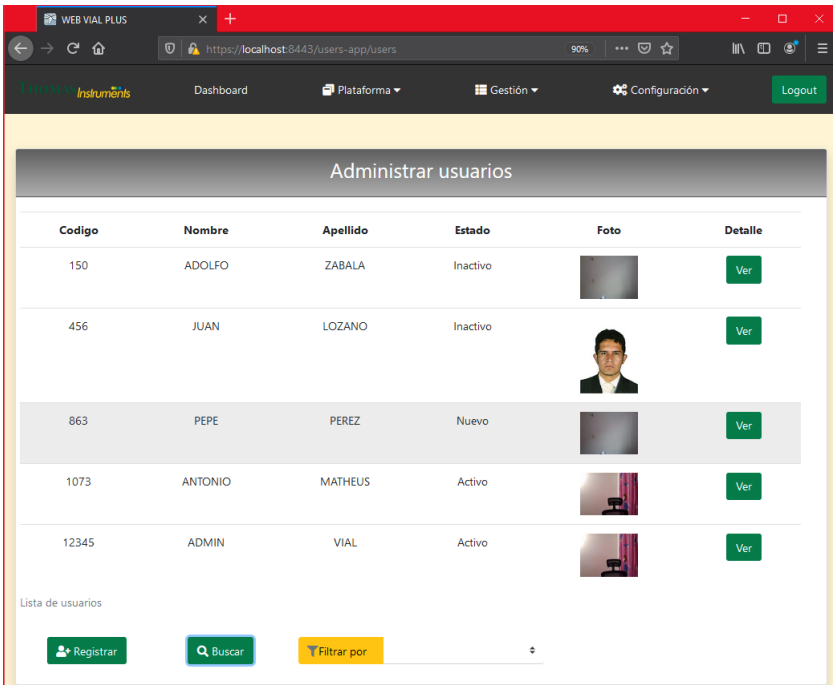
Figura 13

Módulo Administración de Usuarios

Nota. Fuente: elaboración propia.

Figura 14

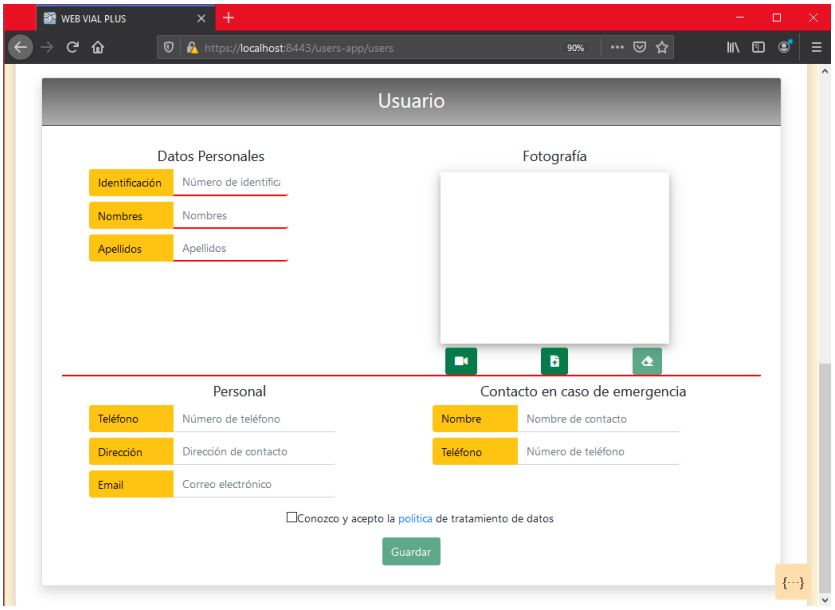
Consulta de Usuarios



Nota. Fuente: elaboración propia.

Figura 15

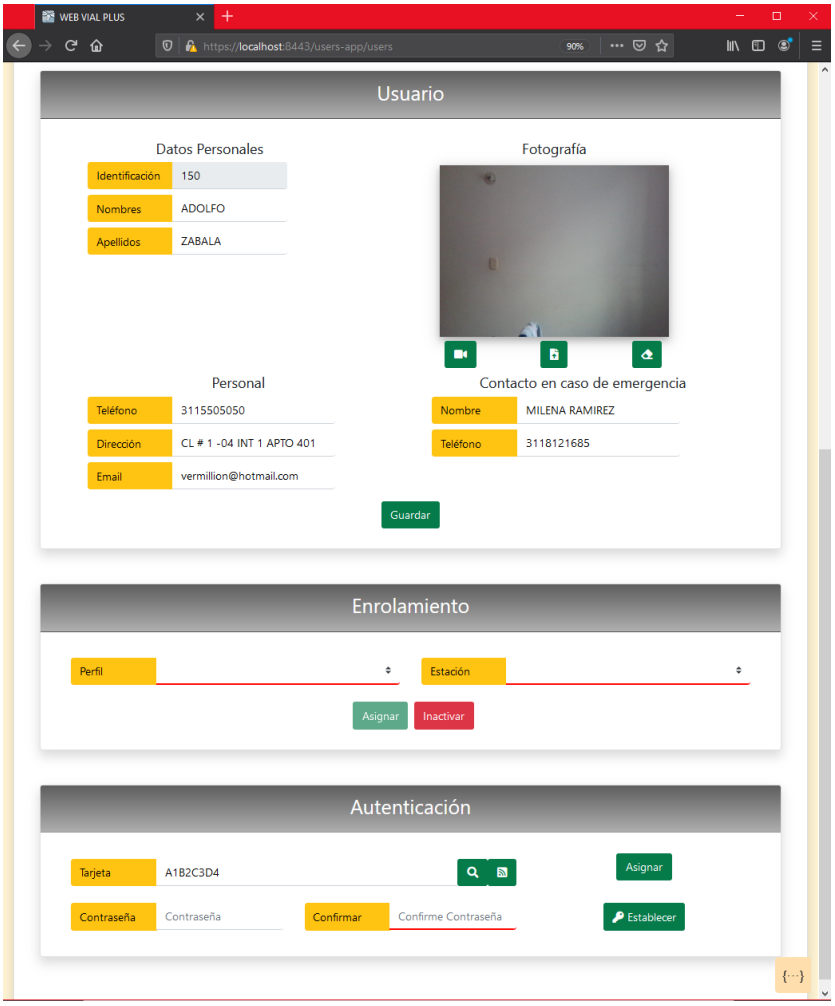
Registro de Usuarios



Nota. Fuente: elaboración propia.

Figura 16

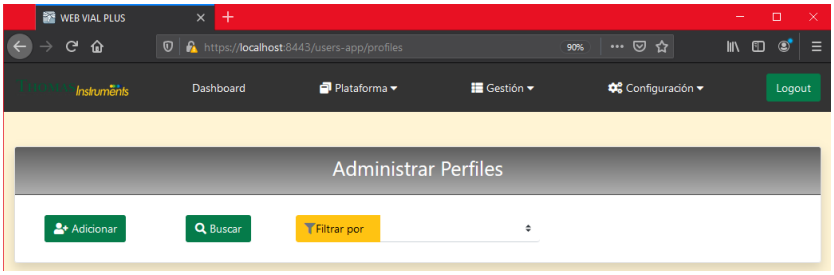
Consulta y Modificación de Usuario



Nota. Fuente: elaboración propia.

Figura 17

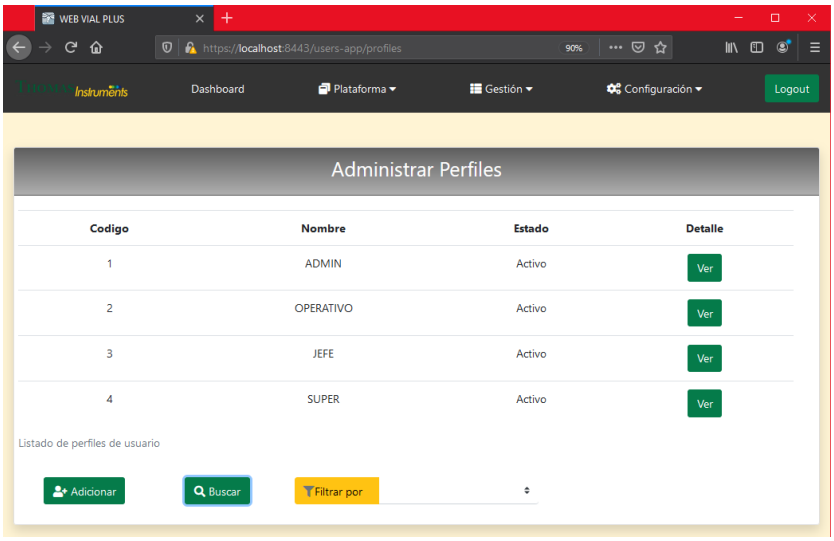
Módulo de Usuarios Administrar Perfiles



Nota. Fuente: elaboración propia.

Figura 18

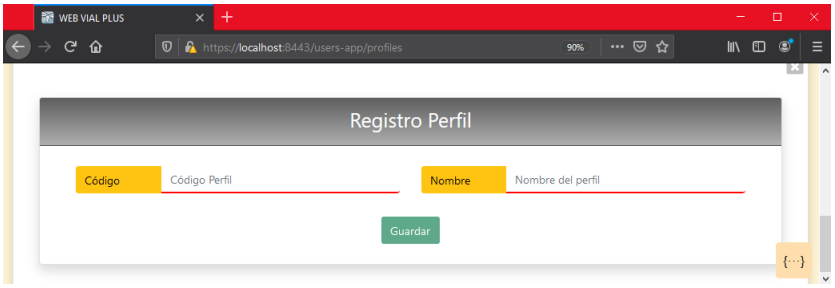
Consulta de Perfiles



Nota. Fuente: elaboración propia.

Figura 19

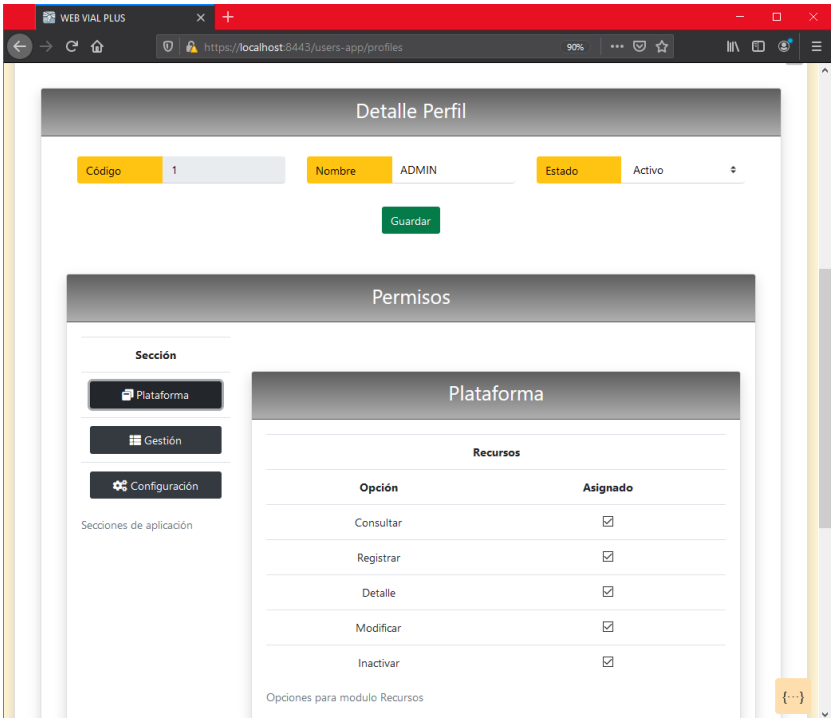
Registro de Perfiles



Nota. Fuente: elaboración propia.

Figura 20

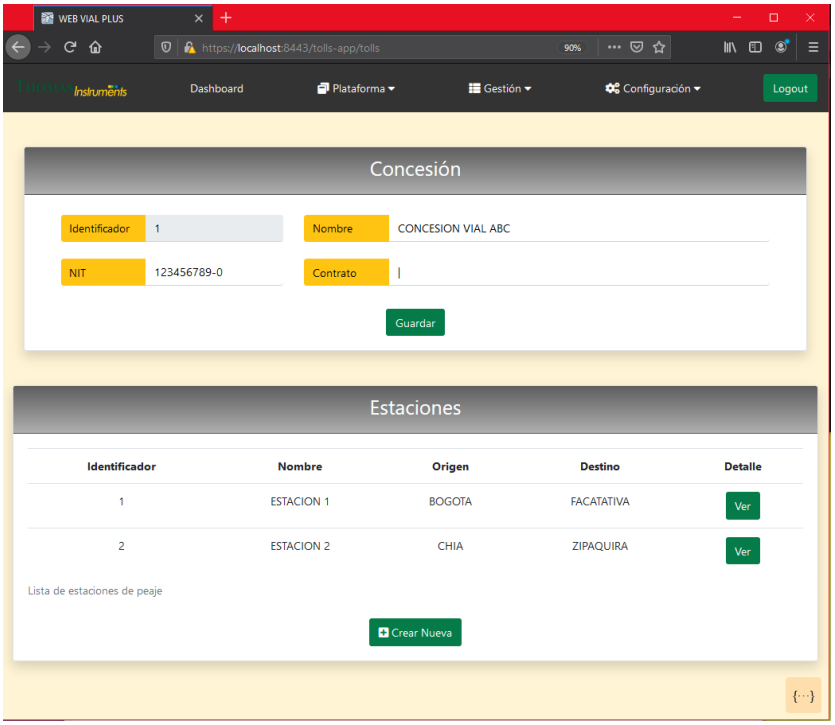
Consulta y Modificación de Perfil



Nota. Fuente: elaboración propia.

Figura 21

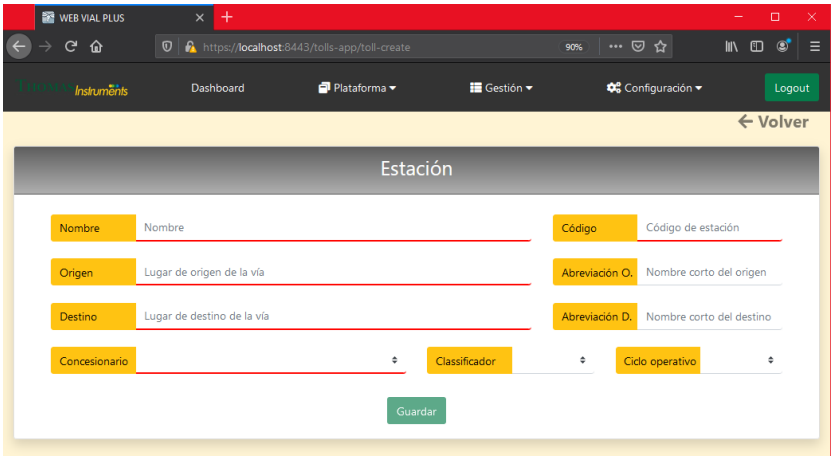
Modulo Configuración de Estaciones



Nota. Fuente: elaboración propia.

Figura 22

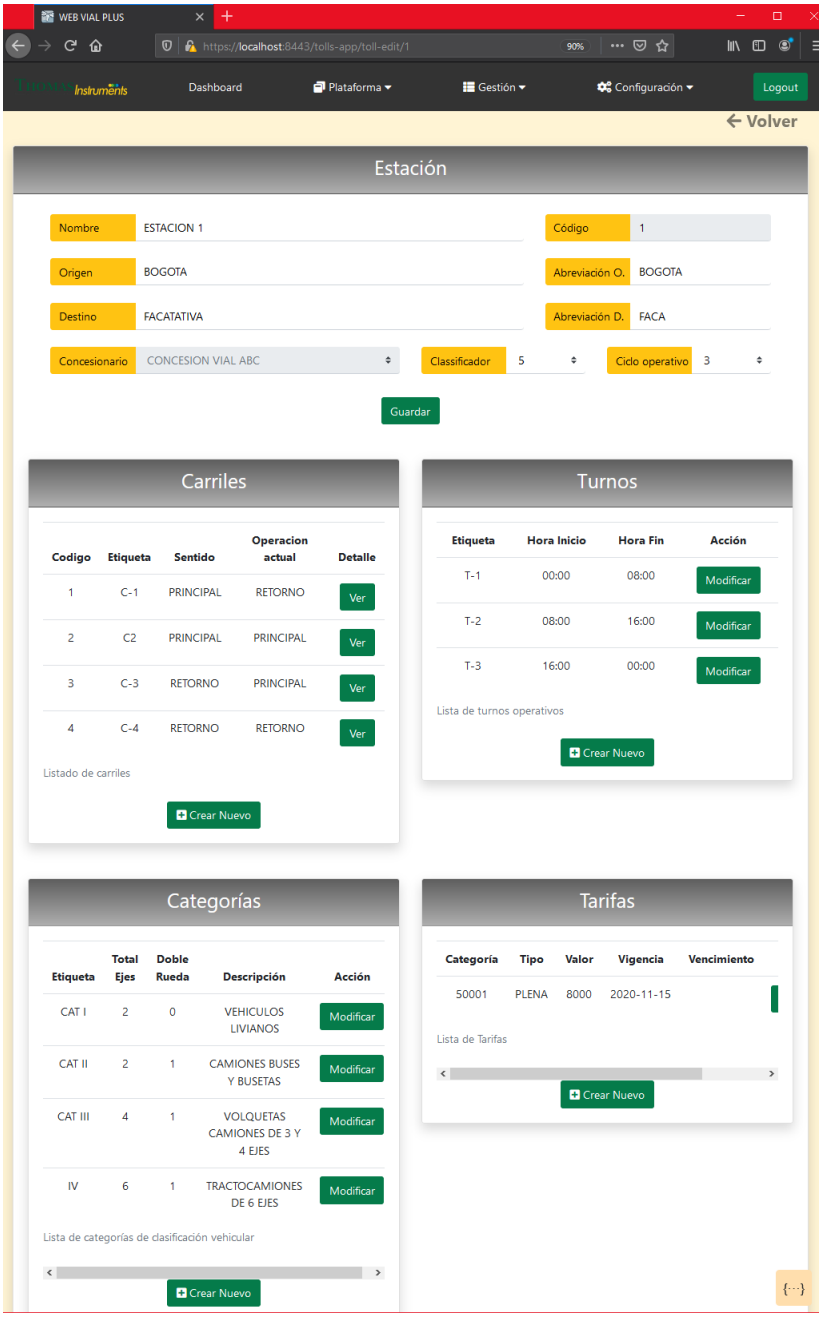
Creación de Estación



Nota. Fuente: elaboración propia.

Figura 23

Consulta y Configuración de Estación



Nota. Fuente: elaboración propia.

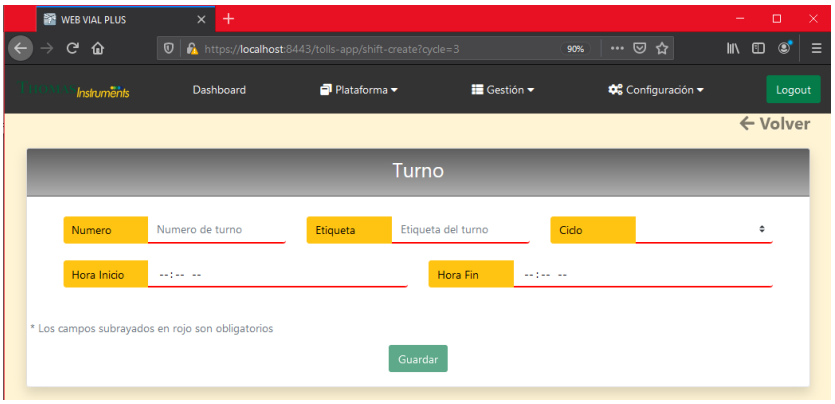
Figura 24

Creación de Carril

Nota. Fuente: elaboración propia.

Figura 25

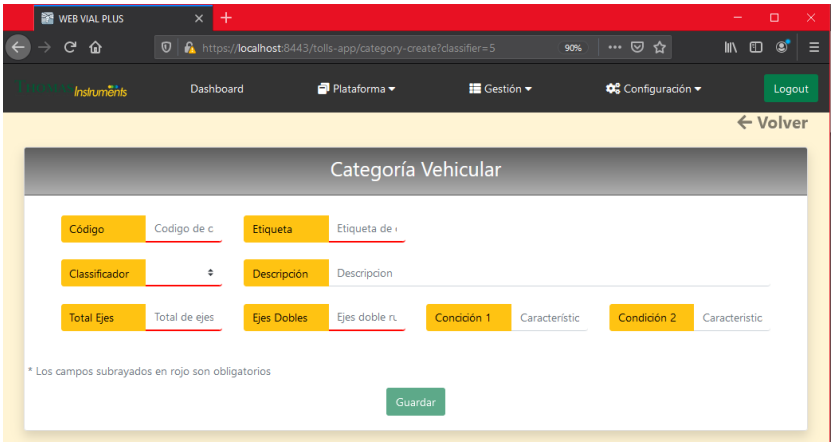
Creación de Turno Operativo



Nota. Fuente: elaboración propia.

Figura 26

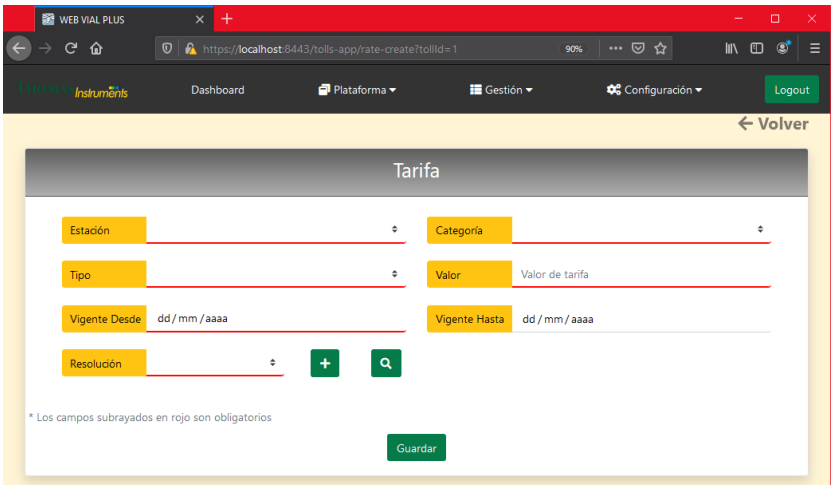
Creación de Categoría Vehicular



Nota. Fuente: elaboración propia.

Figura 27

Creación de Tarifa



Nota. Fuente: elaboración propia.

4.4.1. Herramientas.

En la tabla que se encuentra a continuación, se relacionan las herramientas que se utilizaron en cada una de las diferentes fases del proyecto:

Tabla 6

Herramientas de Construcción

Herramienta	Fase	Tipo	Descripción
Microsoft Planner	Transversal	Aplicación	Gestión y control de actividades.
UML 2.0	Análisis/Diseño	Estándar	Estándar para la definición de los diagramas del proyecto.
Diagrams.net	Análisis	Aplicación	Generación de los diagramas del proyecto.
Open API	Análisis	Estándar	Estándar utilizado para la descripción y definición de las API Rest.
SQL Developer	Diseño /Desarrollo	Aplicación	Ejecución de Scripts de base de datos y generación de diagramas de Modelo Entidad-Relación.
SwaggerHub	Desarrollo	Aplicación	Herramientas para la edición de las API Rest. Permite la generación de código a varios lenguajes a partir de las API Rest definidas.
Java	Desarrollo	Lenguaje	Lenguaje de Implementación.
JavaScript	Desarrollo	Lenguaje	Lenguaje de Implementación.
TypeScript	Desarrollo	Lenguaje	Lenguaje de Implementación.
HTML, CSS	Desarrollo	Lenguaje	Lenguaje de Implementación.
SQL	Desarrollo	Lenguaje	Lenguaje de Implementación.
Boostrap	Desarrollo	Framework	Diseño de las interfaces de usuario.
Spring	Desarrollo	Framework	Implementación de la parte backend de los microservicios.
Angular	Desarrollo	Framework	Implementación de las interfaces de usuario.
Eclipse IDE	Desarrollo	Aplicación	IDE de desarrollo para la parte backend.
VS Code	Desarrollo	Aplicación	IDE de desarrollo para la parte frontend.
Sonar Lint	Desarrollo	Plugin	Complemento utilizado dentro de los IDE (Eclipse y VS Code) para realizar análisis de la

Herramienta	Fase	Tipo	Descripción
			calidad del código en el momento de llevar a cabo la codificación.
Sonar Qube	Desarrollo Pruebas	Aplicación	Herramienta para complementar los análisis de código para el aseguramiento de la calidad del software y presentación de reportes y evidencias.
Single-SPA	Desarrollo	Framework	Herramienta para la aplicación del concepto de microfrontends en la parte de interfaz de usuario.
VisualSVNServer	Desarrollo	Aplicación	Herramienta utilizada para el control de versionado de código.
Maven	Desarrollo	Herramienta	Control de librerías y automatización de procesos de los proyectos de desarrollo.

Nota. Fuente: elaboración propia.

4.4.2. Control de versiones

Para el control de versiones se realizó la organización de las ramas de producción, desarrollo y pruebas, de acuerdo con lo recomendado para la cultura de calidad en lo relacionado a versionado de código.

En la plataforma Visual SVN Server, como herramienta de control de versiones del código, se realiza la inicialización de un repositorio que contiene todos los proyectos de software relacionados a la plataforma Web Vial Plus. Para cada uno de los proyectos de desarrollo, se organizan las ramas de la siguiente forma:

- Producción (master): se establece la carpeta *trunk*, como la ubicación para la rama master o de producción, en la cual se realiza la subida del código y artefactos pasados a producción luego de realizar los procesos de calidad. Cada módulo tiene su proyecto de software se crea la carpeta con el nombre del artefacto Maven del proyecto.
- Desarrollo (develop): en la estructura de carpetas creadas por el servidor se encuentra la carpeta *Braches*; dentro de esta se crea la carpeta *develop*, y allí se crean misma

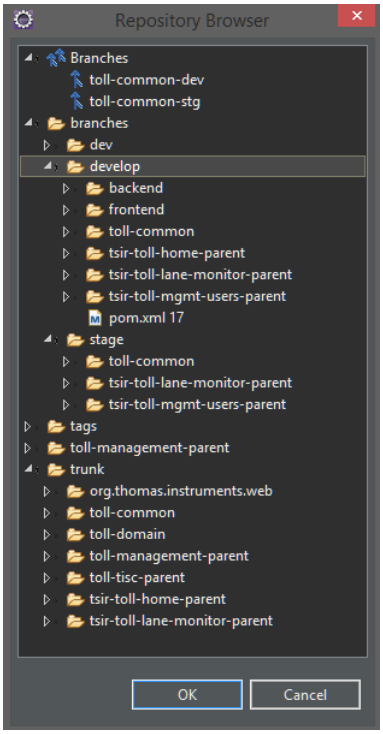
estructura de proyectos establecidas para producción. Los cambios realizados por los desarrolladores deben subir a esta rama.

- Pruebas (stage): en la estructura de carpetas creadas por el servidor se encuentra la carpeta *Braches*; dentro de esta se crea la carpeta *stage*, y allí se crean misma estructura de proyectos establecidas para producción. Los cambios aprobados para pasar a pruebas deben ser subidos por personal con autorización y no ser subidas directamente por los desarrolladores.

En las siguientes imágenes se puede observar la estructuración en el repositorio de control de versiones para el proyecto *thomas-common,* proyecto que contiene las clases de uso transversal y común a todos los proyectos, además de funciones de utilidad; se visualiza con el plugin utilizado en el Eclipse IDE:

Figura 28

Estructuración de Ramas para el Control de Versiones del proyecto toll-common



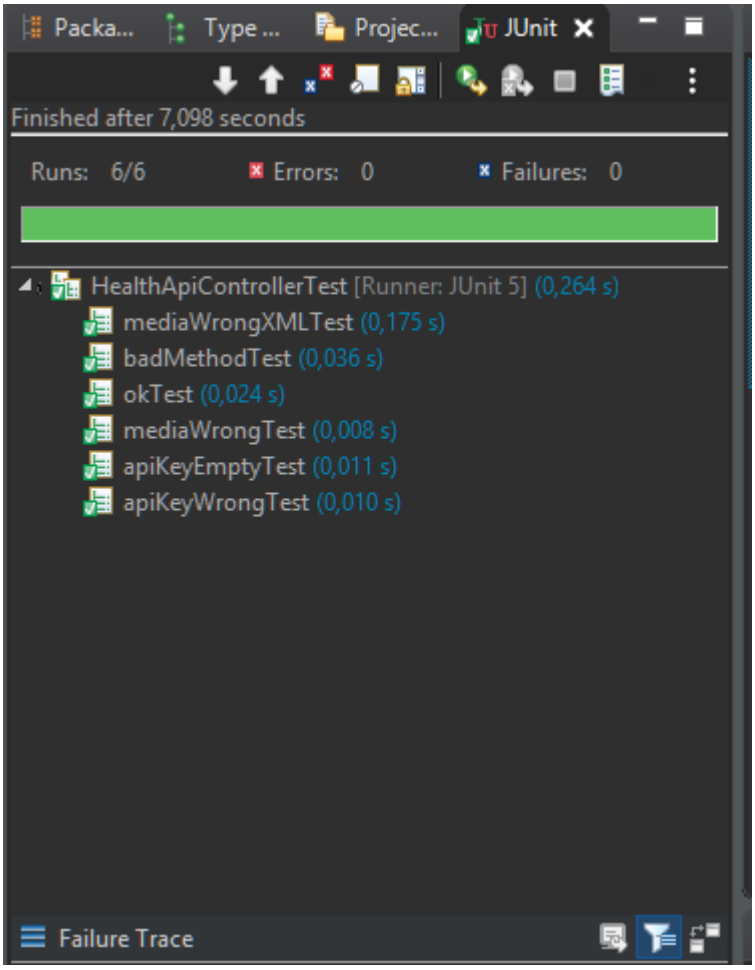
Nota. Fuente: elaboración propia.

4.5. Pruebas

Para el aspecto de pruebas se tienen implementadas pruebas de caja blanca para la validación de los microservicios. Se utiliza el paquete Testing the Web Layer, uno de los componentes del Framework Spring Boot. Esta dependencia permite ejecutar pruebas con basadas en Junit:

Figura 29

Resultados Parciales Ejecución de Pruebas

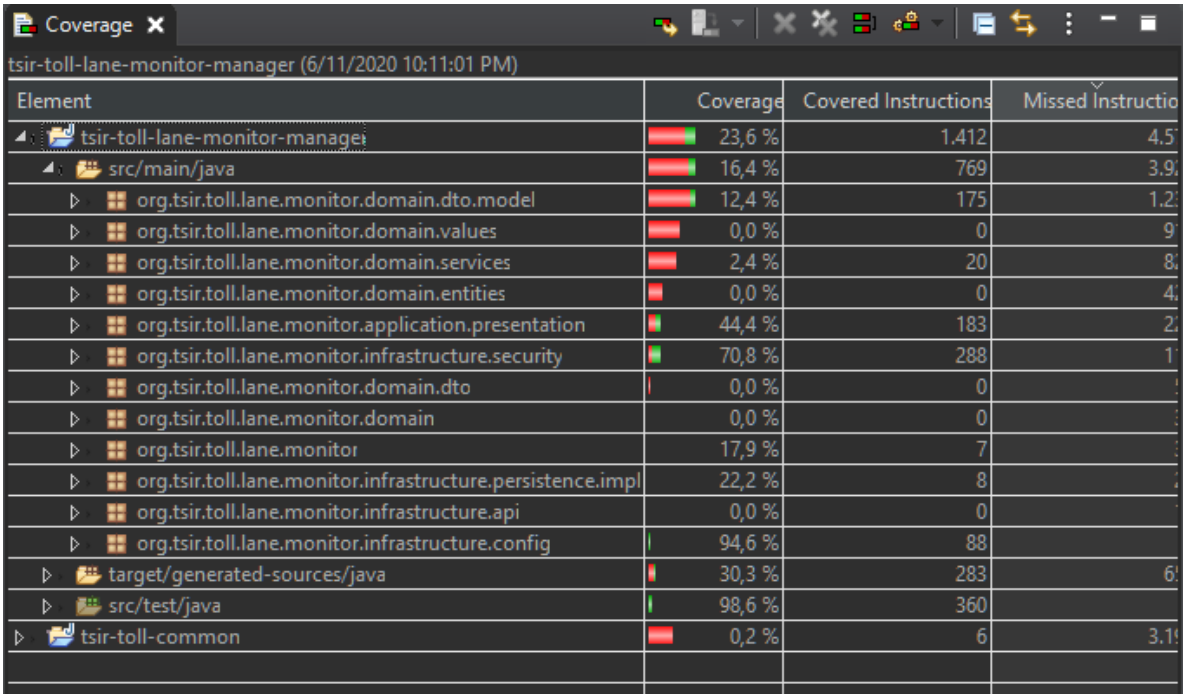


Nota. Fuente: elaboración propia.

Adicionalmente, se complementa con el plugin Coverage de Eclipse IDE, el cual nos permite establecer los valores de cobertura de las pruebas:

Figura 30

Herramienta de Cobertura de Código en Ejecución de Pruebas

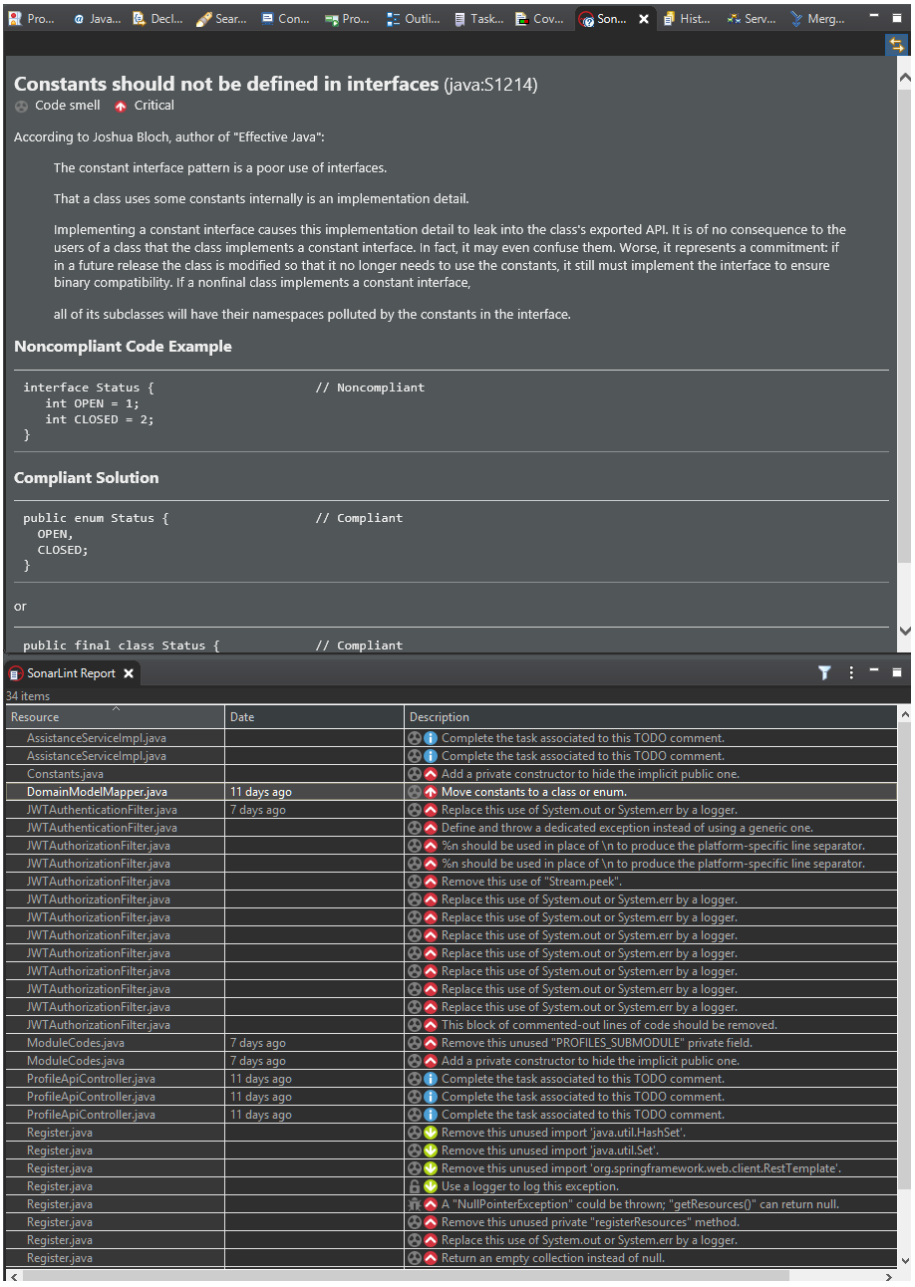


Nota. Fuente: elaboración propia.

Otro aspecto importante es la calidad del código. Para los proyectos se hace uso del plugin SonarLint, el cual permite realizar análisis estático de código.

Figura 31

SonarLint: Herramienta para Análisis de Calidad del Código



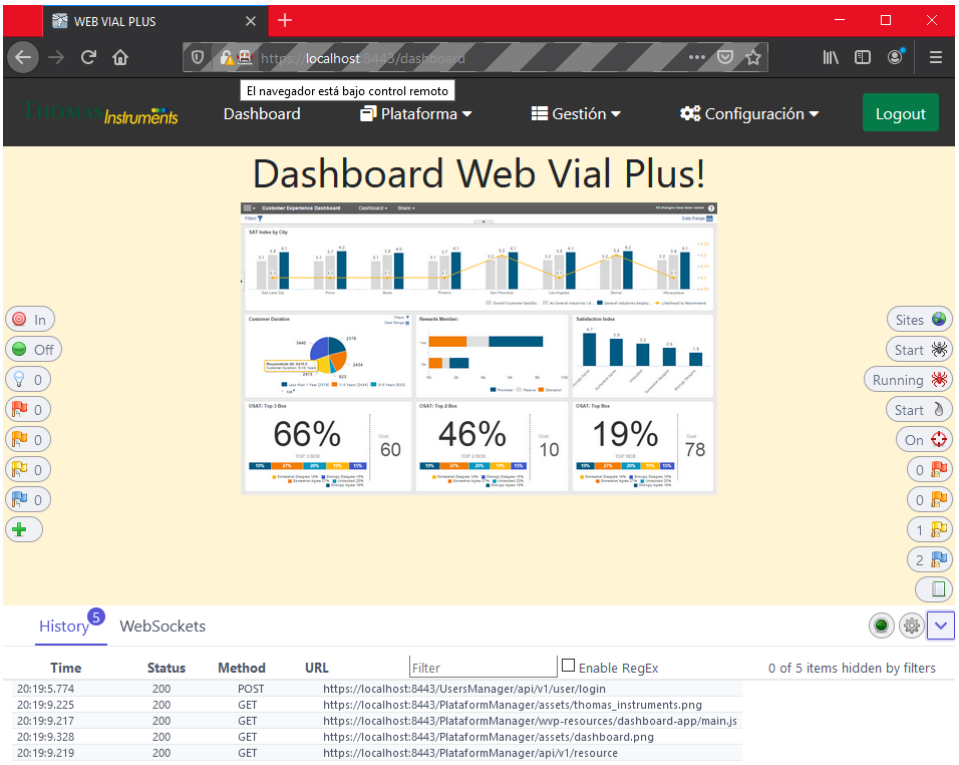
Nota. Fuente: elaboración propia.

Se ejecuta OWAS ZAP versión 2.9.0, con el cual permite ejecutar ataques a la aplicación web desarrollada. Con el complemento que permite utilizar en el navegador Mozilla Firefox

OWAS ZAP como proxy se ejecutan las pruebas de vulnerabilidad, navegando a través de la aplicación:

Figura 32

Ejecución Pruebas de Vulnerabilidades con OWAS ZAP



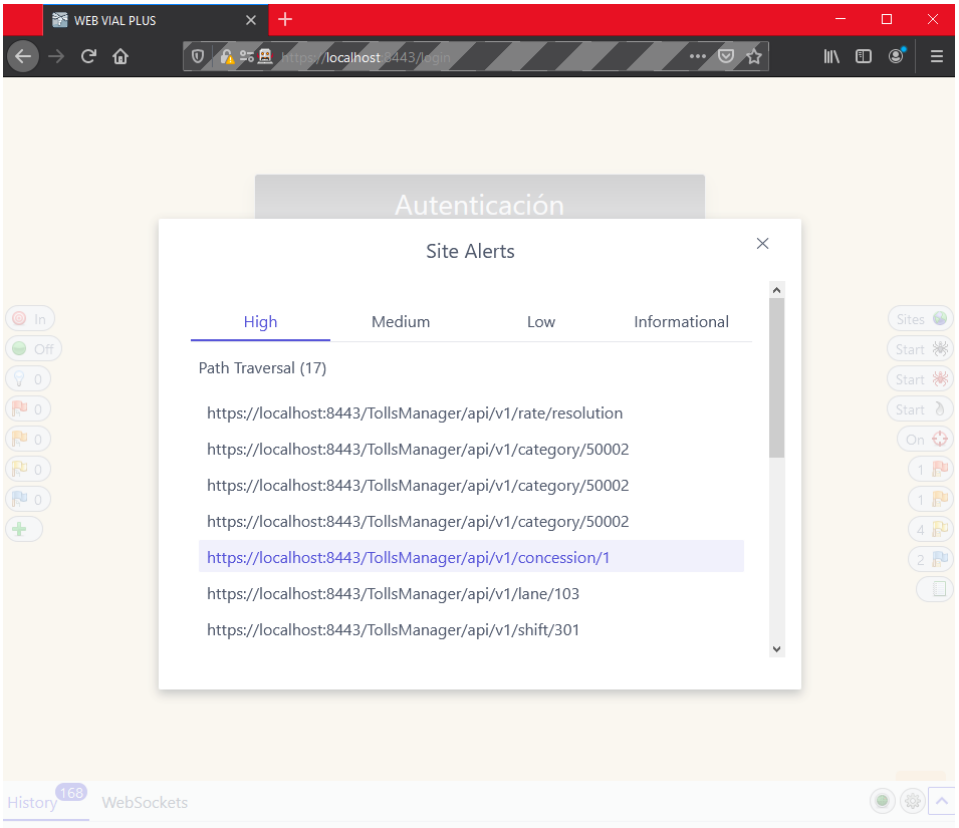
Nota. Fuente: elaboración propia.

Como resultado de la ejecución de las pruebas de vulnerabilidad se presentaron las siguientes vulnerabilidades y se explica su correspondiente acción.

Vulnerabilidad Path Transversal: Mediante esta vulnerabilidad es posible que un atacante pueda obtener o ejecutar información del sistema de archivos, y se detecta que los campos vulnerables son los de tipo cadena de texto:

Figura 33

Identificación Vulnerabilidad Path Transversal

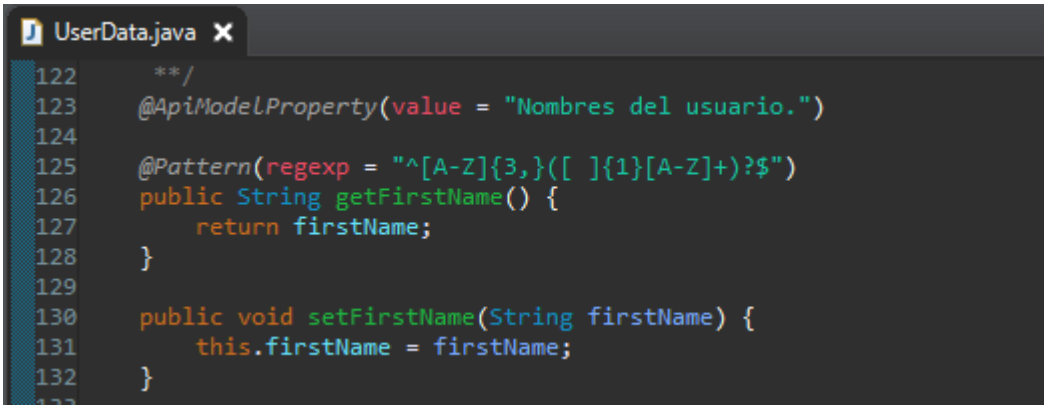


Nota. Fuente: elaboración propia.

Acción para Path Transversal: se realiza inicialmente la restricción de los campos en el lado del backup (se realizan principalmente en el servidor debido a que como microservicio estará expuesto a diversos clientes y no solo a la parte del frontend), con validaciones aplicando patrones. Por ejemplo, en la siguiente imagen, se establece que el nombre del usuario solo puede contener una o dos palabras (separadas con espacio) con una longitud mínima de tres caracteres, con solo caracteres alfabéticos:

Figura 34

Validaciones de Campos al Lado del Backend



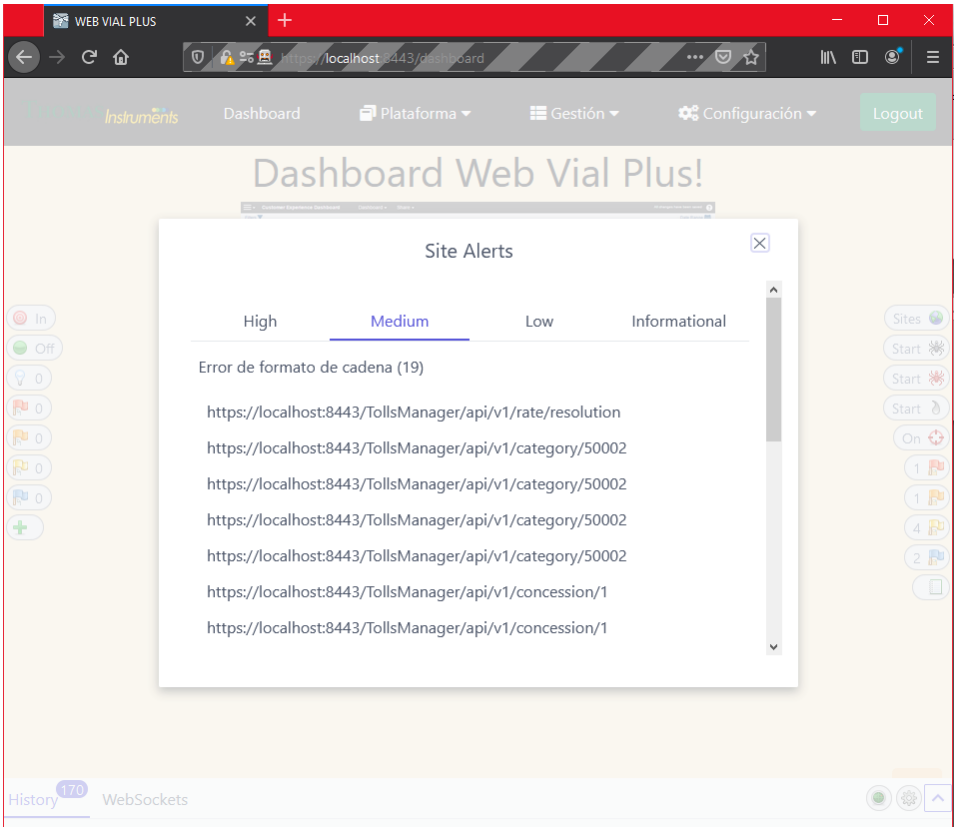
Nota. Fuente: elaboración propia.

En el momento los requerimientos identificados se pueden manejar con este tipo de validaciones, es decir, que los campos de tipo texto, pueden ser validados como cadenas que solo contiene cadenas con caracteres alfabético y con espacios. Para campos como dirección solo se permitirán los caracteres '#' y '-'.

Vulnerabilidad Format string attack: esta vulnerabilidad ocurre cuando los datos de una cadena de entrada son evaluados como un comando por la aplicación, por ejemplo con patrones de formateo de una cadena, y se detecta que los campos vulnerables son los de tipo cadena de texto

Figura 35

Identificación Vulnerabilidad Format String Attack



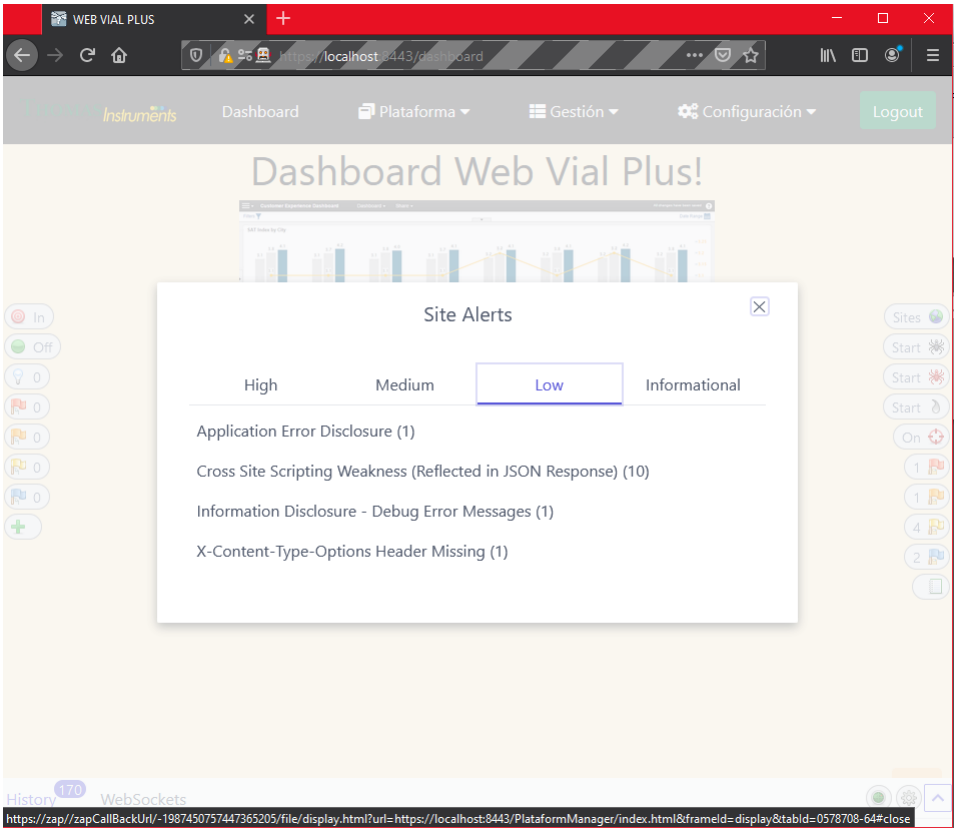
Nota. Fuente: elaboración propia.

Acción para Format string attack: se realiza inicialmente la restricción de los campos en el lado del backup (se realizan principalmente en el servidor debido a que como microservicio estará expuesto a diversos clientes y no solo a la parte del frontend), con validaciones aplicando patrones igualmente que con la anterior vulnerabilidad de Path Transversal.

Finalmente se identifican cuatro vulnerabilidades de impacto bajo: *Application Error Disclosure, Cross Site Scripting Weaknees, Information Disclosure* y *X-Content-Type-Options header missing:*

Figura 36

Vulnerabilidades de Impacto Bajo



Nota. Fuente: elaboración propia.

Acción para Application Error Disclosure y *Information Disclosure*: se detecta que no se encuentra controlado un error para la conversión de una cadena a fechas y se está enviando la traza del error en la respuesta al cliente, la cual puede proveer información de los componentes o librerías con las que está desarrollada la aplicación, exponiendo información de componentes a un atacante. Se procede controlar el error mediante la captura de la excepción y enviando un mensaje de error estandarizado para las API definidas con el texto de ‘Formato de fecha errado’.

Para la vulnerabilidad de *Cross Site Scripting*, se encuentra que no se están validando la información de los campos de consulta en los filtros que reciben valores alfanuméricos. Se realizan una validación de los valores de acuerdo con el tipo de filtro seleccionado.

Adicionalmente también se ejecuta la herramienta SonarQube para establecer la calidad del código, verificación de vulnerabilidades de la aplicación y adicionalmente también establece Security Hotspots (puntos calientes de seguridad), lo cuales son puntos que deben ser revisados manualmente para establecer si son o no vulnerabilidades de seguridad. Se realiza una breve descripción de los resultados en el siguiente segmento con la visualización de las correspondientes imágenes de cada uno de los diferentes proyectos.

Módulos Frontend: Con respecto a aspectos de seguridad no se encuentran vulnerabilidades en los proyectos. Los puntos por verificar en los hotspots reportados corresponden a la recomendación de actualizar un componente para el procesamiento de patrones de cadenas, el cual se realizará posteriormente. Se observa una aceptable deuda técnica por bugs y duplicaciones de código, presentado por la inexperiencia con el lenguaje de desarrollo. Como acción para la reducción de deuda técnica se procederá a crear un proyecto de utilidades que pueda contener lo componentes reutilizables transversales a todos los proyectos del frontend.

Módulos Backend: No se reportan vulnerabilidades en los proyectos. La revision de hotspots corresponden a:

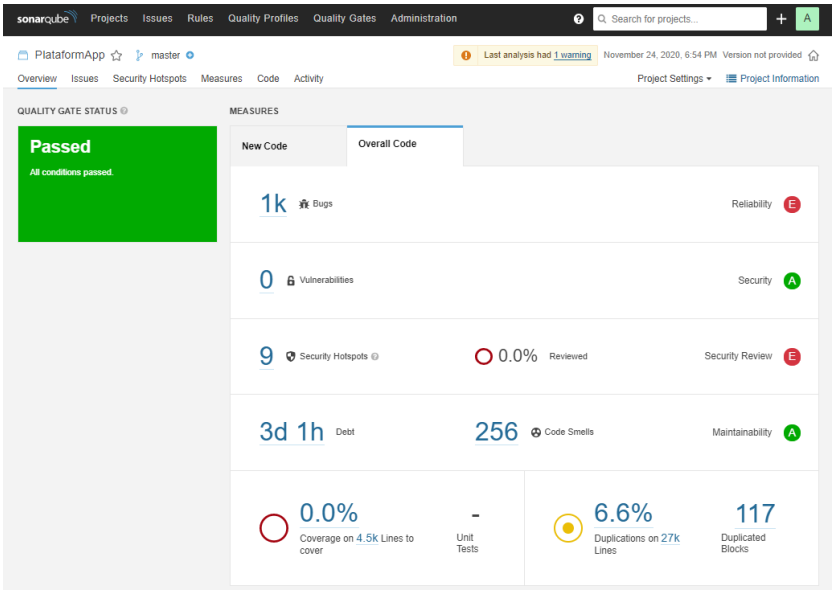
a) CSRF Deshabilitado: se da por aceptado pues no se requiere implementarlo en microservicios por ser stateless, y el control de autenticación y autorización es gestionado con JWT.

b) CORS Habilitado: se establece debido a la necesidad de exposición de las API a clientes externos y no solamente por la parte del frontend. Se evaluará para mitigar esta vulnerabilidad la implementación del API Gateway.
c) Verificación de control de permisos de acceso a los endpoints expuestos de los controladores de las API Rest: Se realiza la verificación y todos los endpoints están cubiertos correctamente con su correspondiente cadena de autorización.

En estos proyectos se observa baja deuda técnica gracias a la experiencia de desarrollo con el lenguaje Java.

Figura 37

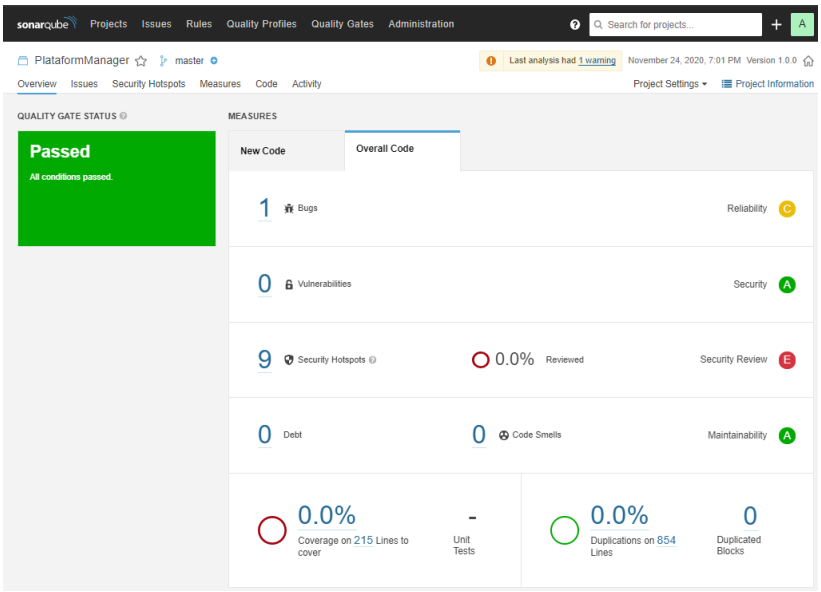
Análisis SonarQube Módulo Plataforma (Frontend)



Nota. Fuente: elaboración propia.

Figura 38

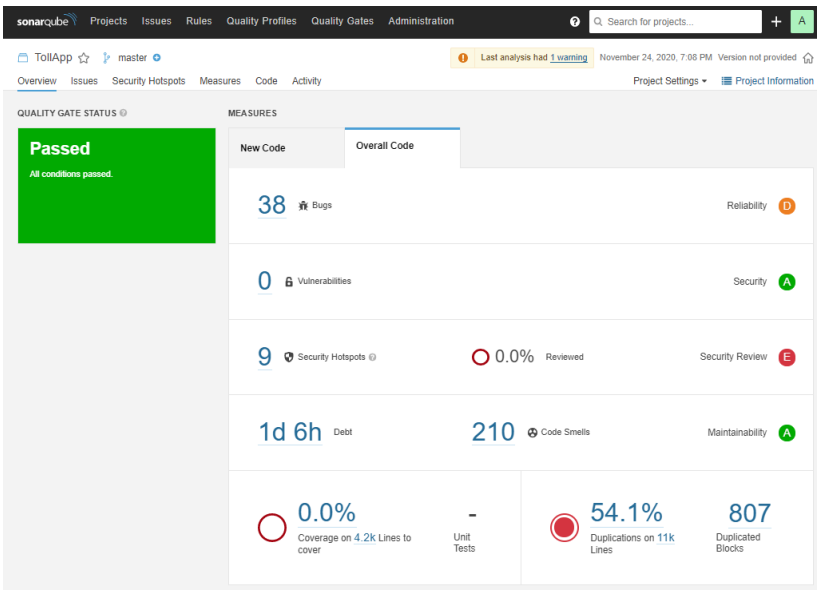
Análisis SonarQube Módulo Plataforma (Backend)



Nota. Fuente: elaboración propia.

Figura 39

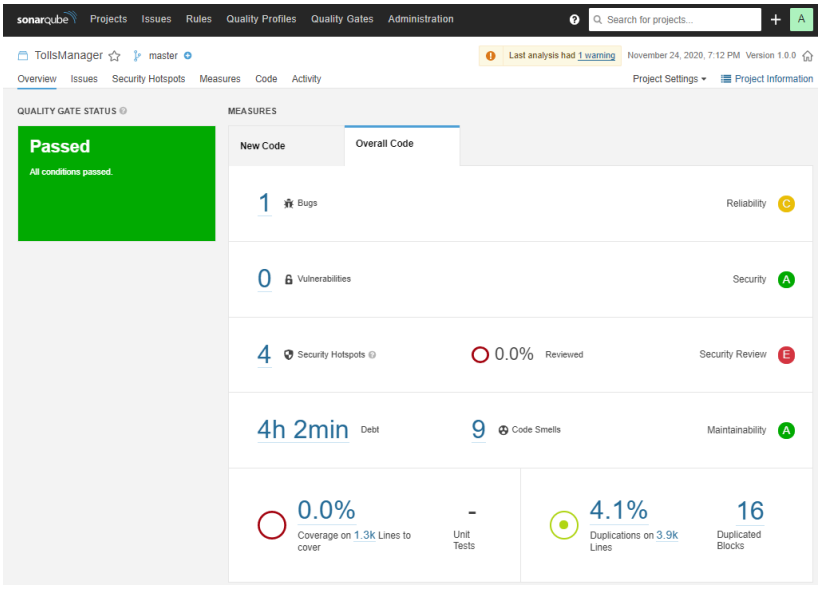
Análisis SonarQube Módulo Estación (Frontend)



Nota. Fuente: elaboración propia.

Figura 40

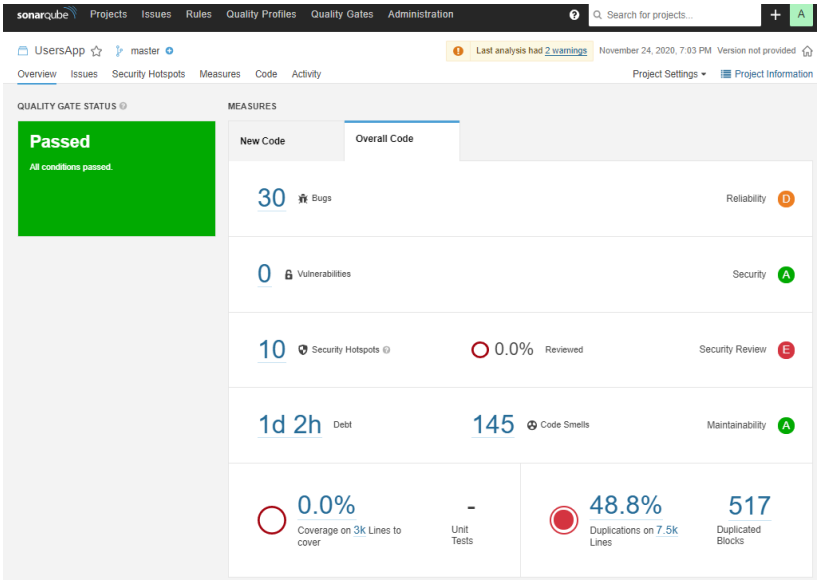
Análisis SonarQube Módulo Estación (Backend)



Nota. Fuente: elaboración propia.

Figura 41

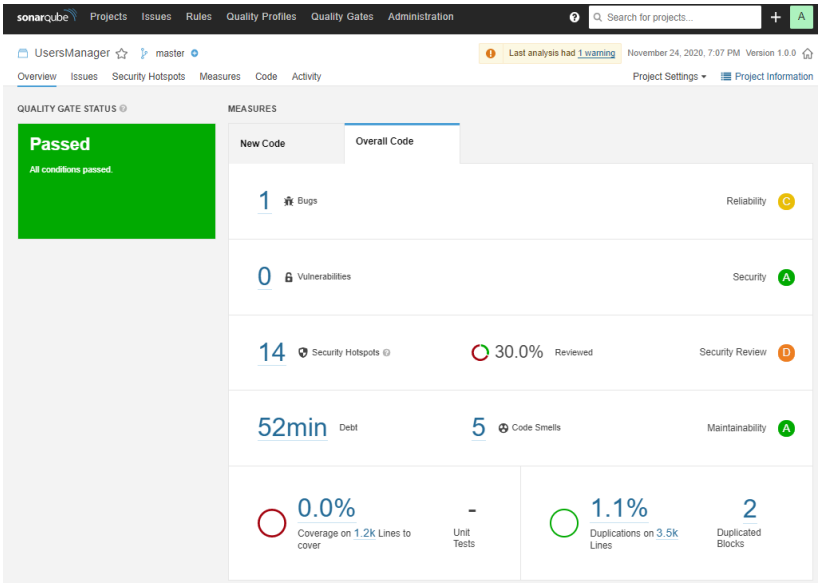
Análisis SonarQube Módulo Usuarios (Frontend)



Nota. Fuente: elaboración propia.

Figura 42

Análisis SonarQube Módulo Usuarios (Backend)



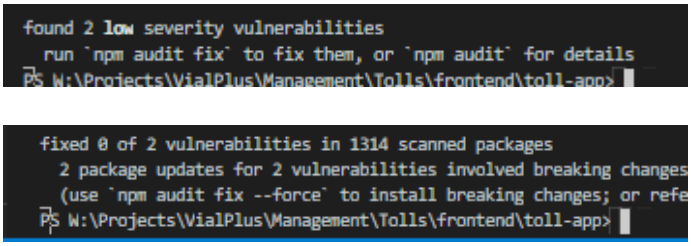
Nota. Fuente: elaboración propia.

Para los proyectos del Frontend desarrollados en Angular, se ejecuta como herramienta de identificación de vulnerabilidades de paquetes el comando de Node *npm audit,* que permite identificar los componentes de terceros utilizados y que se encuentren desactualizados o con vulnerabilidades, adicionalmente al establecer la opción adicional *fix* permite realizar una actualización de los paquetes que presentan vulnerabilidades. Luego de la ejecución de la opción *fix,* quedan algunos componentes que no es posible actualizar debido a que implican cambios fuertes, y pueden afectar la composición del proyecto. Para el actual proyecto quedan identificados para una posterior etapa con el criterio que las vulnerabilidades dejadas son de baja severidad.

En el módulo de Configuración de Estación la herramienta identificó inicialmente 2 vulnerabilidades, las cuales no fue posible su solución inmediata, como se observa en la siguiente imagen:

Figura 43

Componentes Vulnerables del Módulo Estación (Frontend)

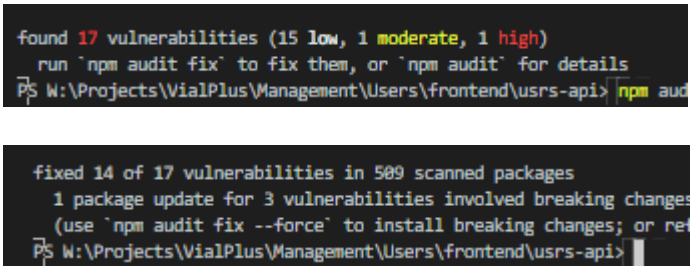


Nota. Fuente: elaboración propia.

En el módulo de Gestión de Usuarios la herramienta identificó inicialmente 17 vulnerabilidades, las cuales no fue posible la solución inmediata de 3 vulnerabilidades, como se observa en la siguiente imagen:

Figura 44

Componentes Vulnerables del Módulo Usuarios (Frontend)

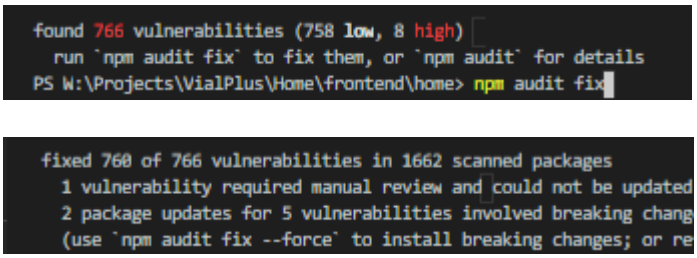


Nota. Fuente: elaboración propia.

En el módulo de Administración de Plataforma, la herramienta identificó inicialmente 766 vulnerabilidades, las cuales no fue posible la solución inmediata de 6 vulnerabilidades, como se observa en la siguiente imagen:

Figura 45

Componentes Vulnerables del Módulo Plataforma (Frontend)



Nota. Fuente: elaboración propia.

4.6. Instalación y Configuración

A continuación, se relaciona el proceso para realizar la instalación y configuración de los módulos necesarios para la plataforma Web Vial Plus.

4.6.1. Servidor de Aplicaciones WildFly

El servidor seleccionado para el despliegue de los módulos es el servidor de aplicaciones WildFly.

La instalación del aplicativo es simple y consiste en descomprimir el contenido del distribuible en la carpeta seleccionada para su instalación. Los siguientes pasos deben ser realizados cuando se realiza una instalación del servidor con un distribuible descargado directamente de su sitio web o para una verificación de una instalación de un distribuible de otro origen.

El servidor cuenta con dos modos de ejecución: standalone y domain; el requerido para el ambiente de desarrollo es el standalone con el perfil por defecto (default). Para este modo de ejecución, algunas configuraciones deben ser establecidas por el archivo de configuración standalone.xml, encontrado en la carpeta WILDFLY_HOME/standalone/configuration.

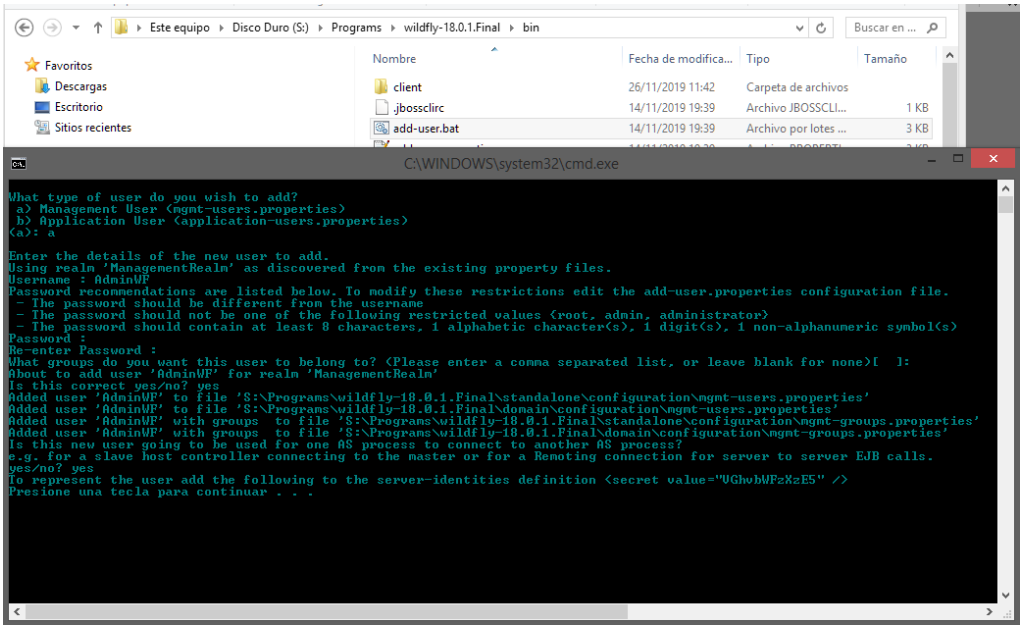
Crear usuario administrador.

Se debe realizar la creación de un usuario de administración. Para el entorno de desarrollo este usuario es suficiente, permite realizar cambios en la configuración del servidor y acceder a la sección de despliegues.

Se debe ejecutar el archivo add-user.bat encontrado en la ruta WILDFLY_HOME/bin y seguir las instrucciones de la ejecución como se visualiza en la siguiente imagen:

Figura 46

WildFly Creación Usuario Administrador



Nota. Fuente: elaboración propia.

Como estándar para el ambiente de desarrollo se recomienda utilizar como nombre de usuario AdminWF y contraseña Thomas_19. Tanto usuario y contraseña son case sensitive.

Configurar interfaces de red.

Por defecto, el servidor de aplicaciones viene configurado solo para ingresar mediante la dirección IP local (127.0.0.1 o localhost). Para permitirnos acceder mediante el segmento de red LAN debemos habilitar las interfaces editando el archivo de configuración standalone.xml:

Figura 47

Configuración Interfaces de Red WildFly

```
<interfaces>
    <interface name="management">
        <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
    </interface>
    <interface name="public">
        <any-address/>
    </interface>
</interfaces>
```

Nota. Fuente: elaboración propia.

Crear Datasource.

Para crear un Datasource (recurso de conexión a base de datos), se realiza a través de la interfaz de configuración por la línea de comandos, encontrado en la ruta WILDFLY_HOME/bin.

Se requiere tener descargado el controlador JDBC del motor de la base de datos a utilizar. Por ejemplo, para la base de datos Oracle el controlador es el OJDBC.

Acceder mediante línea de comando a la ruta WILDFLY_HOME/bin y ejecutar los siguientes comandos:

```
jboss-cli.bat --connect controller=127.0.0.1

module add --name=com.oracle --resources="
S:\Software\JavaAPI\JDBC\Oracle\12.2.0.1\ojdbc8.jar" --
dependencies=javax.api,javax.transaction.api

/subsystem=datasources/jdbc-driver=oracle:add(driver-
name="oracle",driver-module-name="com.oracle",driver-class-
name=oracle.jdbc.driver.OracleDriver)

data-source add --jndi-name=java:jboss/TollDS --name=TollDS --connection-
url=jdbc:oracle:thin:@localhost:1521:xe --driver-name=oracle --user-
name=tjavao --password=admin
```

El jndi-name es el nombre como se identificará el recurso para referenciarlo en las aplicaciones o módulos desarrollados. Se deben ajustar la url, usuario y contraseña de acuerdo con el ambiente asignado.

Habilitar modo seguro – Auto firmado.

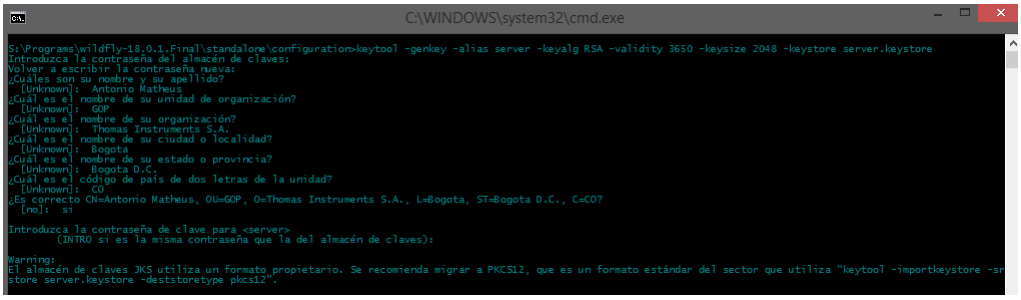
Para habilitar la conexión con transporte seguro para los módulos y APIs de la Web vial Plus, mediante por https con protocolo TLSv1.2 en el ambiente de desarrollo, se acudirá a un certificado auto firmado almacenado en un almacén de claves; por seguridad en producción se deberá instalar un certificado firmado por una entidad válida.

Realizar los siguientes pasos para habilitar esta característica:

a- Crear un almacén de claves: El JDK de Java tiene la utilidad para realizar la creación del almacén de claves:

Figura 48

Creación Almacén de Claves.



Nota. Fuente: elaboración propia.

b- Ejecutar los siguientes comandos por CMD:

```
#jboss-cli.bat --connect controller=127.0.0.1

/subsystem=undertow/server=default-server/https-listener=https:read-
attribute(name=security-realm)

batch

/subsystem=elytron/key-
store=ThomasKeyStore:add(path=server.keystore,relative-
to=jboss.server.config.dir,credential-reference={clear-
text=Thomas_20},type=JKS)
```

```
   /subsystem=elytron/key-manager=ThomasKeyManager:add(key-
store=ThomasKeyStore,credential-reference={clear-text=Thomas_20})
   /subsystem=elytron/server-ssl-context=ThomasSSLContext:add(key-
manager=ThomasKeyManager,protocols=["TLSv1.2"])
   run-batch
   reload
```

Despliegue de los módulos.

Los módulos de la aplicación Web Vial Plus, son liberados como archivos WAR. EL servidor de aplicaciones WildFly tiene la opción de hacer el despliegue automáticamente mediante escaneo de la carpeta de despliegues, que se encuentra en la ruta *WILDFLY_HOME/standalone/deployments*, por lo anterior, se deben copiar los archivos de los módulos en esta ubicación y se realiza el despliegue de forma automática.

Entendiendo que la forma de despliegue, explicada anteriormente conlleva a riesgos de seguridad, en una siguiente fase, se le realizarán varias mejoras a este proceso de instalación.

5. Anexos

5.1. Documento de Arquitectura de Software

5.2. Planteamiento metodología de seguridad

5.3. Aspectos de Gobierno estratégico

6. Conclusiones

La falta de definición de una arquitectura de software en un proyecto tarde o temprano pueden llevar al fracaso de una aplicación, tal como se pudo identificar en el software Web Vial, en el cual, los esfuerzos se han orientado al mantenimiento de difícil ejecución debido a la deuda técnica de la misma y han detenido el crecimiento de la aplicación.

El seguir estándares y lineamientos para el desarrollo de un proyecto, hacer que se pueda llevar a cabo de una forma eficiente y efectiva y al establecer todas estas condiciones en un documento como el Documento de Arquitectura, nos permite avanzar en el desarrollo de un producto con calidad.

Son muchos los aspectos para tener en cuenta a la hora de desarrollar un proyecto de software, no solo es enfocarse en las funciones que queremos realizar en la plataforma, sino que también debemos pensar en calidad, seguridad rendimiento y otros tantos aspectos que podemos identificar y priorizar con la arquitectura de software. Uno de los aspectos importantes hoy en día es la seguridad, por el aspecto sensible que se da a la información. En la plataforma actual no se le había dado relevancia a la seguridad aun cuando los procesos tienen que ver con aspectos que representan dinero a los clientes, y claramente a la misma compañía.

La modularidad que se busca con el proyecto, impactará de forma positiva a los atributos de negocio, adicional de aplicar metodologías ágiles para el proceso.

Lo desarrollado en el presente proyecto debe seguir evolucionando en próximas etapas para la búsqueda de implementación de procesos de Integración y Entrega continua, y se buscó dejar una excelente base para que la empresa de continuidad hacia estos aspectos.

7. Referencias

Convial. (s.f.). *Brochure_peajes.indd - peajes.pdf*. Obtenido de http://www.convial.net/peajes.pdf

i+D3. (s.f.). *3Toll Sistema de gestión de peajes*. Obtenido de https://imasdetres.com/mx/trafico/sistemas/3toll-sistema-de-control-de-peajes/nivel-de-area/

ISACA. (2012). *COBIT® 5: Procesos Catalizadores.* Rolling Meadows, IL EE.UU.: ISACA.

SICE. (s.f.). *Sistemas de Peaje | SICE*. Obtenido de https://www.sice.com/sites/Sice/files/2016-12/TI_PEAJES.pdf

Tecsidel. (s.f.). *Product-Sheet-Free-Flow-1.pdf*. Obtenido de https://tecsidel.com/wp-content/uploads/2018/02/Product-Sheet-Free-Flow-1.pdf

auth0.com. (s. f.). *JWT.IO - JSON Web Tokens Introduction*. JSON Web Token. Recuperado 6 de noviembre de 2020, de http://jwt.io/

Clements, P. C. (1996). A survey of architecture description languages. *Proceedings of the 8th International Workshop on Software Specification and Design*, 16-25. https://doi.org/10.1109/IWSSD.1996.501143

Cohn, M. (2009). En *User Stories Applied* (p. 4). Pearson Education Inc. http://athena.ecs.csus.edu/~buckley/CSc191/User-Stories-Applied-Mike-Cohn.pdf

Fowler, M., & Lewis, J. (2014). *Microservices*. martinfowler.com. https://martinfowler.com/articles/microservices.html

Jackson, C. (2019, junio 19). *Micro Frontends*. Martinfowler.Com. https://martinfowler.com/articles/micro-frontends.html

Microsoft. (s. f.). *Microsoft Security Development Lifecycle*. Recuperado 6 de noviembre de 2020, de https://www.microsoft.com/en-us/securityengineering/sdl

Somerville, I. (2005). Ingeniería del Software. En *Ingenieía del software* (7.ª ed., p. 6). Pearson Education S.A.

Documento de arquitectura de Software.

Web Vial Plus

Versión 1.0

CONTROL DE VERSIONES

Versión	Cambios	Fecha	Elaborado	Aspectos Conceptuales
1.0	Creación del documento.	22/08/2020	Antonio Matheus	N/A
1.1	Actualización general de diagramas.	01/10/2020	Antonio Matheus	N/A
1.2	Complemento sección Guías de desarrollo.	03/11/2020	Antonio Matheus	N/A

CONTENIDO

INTRODUCCION.

El presente documento presenta la descripción arquitectural del sistema Web Vial Plus, y las decisiones relevantes para las interacciones de los diferentes componentes del sistema. Adicionalmente se presentan los estándares y guías que se deben seguir al momento de realizar el desarrollo de los diferentes componentes que hacen parte de la aplicación.

La solución resultante es un conjunto de servicios que se despliegan en un contenedor de aplicaciones y conteniendo las funcionalidades.

El documento se encuentra organizado en cuatro secciones: la primera sección describe la finalidad del documento y lo requerido para entender su contenido. La segunda sección contiene la representación de la arquitectura a alto nivel, en el cual se presentan los diagramas con una notación propia, pero lo suficientemente clara para obtener una vista general del sistema a desarrollar. En la tercera sección se realiza la descripción más detallada de los módulos y componente de la aplicación haciendo uso de diferentes vistas junto con sus correspondientes diagramas en notación UML. En la cuarta y última sección se presentan las decisiones relacionadas con los requisitos no funcionales y guías para unificar criterios de desarrollo de la aplicación.

Propósito.

Realizar la definición de las diferentes vistas para llegar a un entendimiento y comprensión clara del sistema de software desarrollado. Las vistas se presentan siguiendo el modelo de arquitectura 4+1, con el objetivo de presentar la descripción del sistema a diferentes tipos de interesados y el proceso de ciclo de vida de la aplicación.

Alcance.

Se presentan las siguientes vistas, los diagramas utilizados y una breve descripción de lo que representa cada una de ellas:

Vistas	Diagramas UML	Descripción
ESCENARIOS	Casos de Uso	Funcionalidades del sistema
LÓGICA	Modelo Entidad-Relación	Objetos
DESARROLLO	Componentes	
FÍSICA	Despliegue	
PROCESOS	Clases, Secuencia	

Otro factor para tener en cuenta, el que se presenta la descripción del módulo principal y dos módulos desarrollados.

Audiencia.

El documento contiene las vistas necesarias para entender el sistema desde el punto de vista de los diferentes interesados en el ciclo de vida del software.

En la siguiente tabla se relacionan los interesados identificados, el tipo de proceso que influye y la vista o vistas establecidas para el entendimiento del proceso relacionado:

Interesados	Proceso	Vista/Vistas
Gerente de operaciones	Operación	Escenarios

Director de operaciones		
Director de proyectos		
Clientes finales		
Área de soporte	Operación, Implementación	Escenarios, Física
Área de análisis y diseño	Análisis y Diseño	Escenarios, Lógica, Procesos
Área de desarrollo	Desarrollo	Escenarios, Lógica, Desarrollo
Área de pruebas	Pruebas	Escenarios, Lógica
Área de redes e infraestructura	Implementación	Física

Definiciones, Acrónimos y abreviaciones.

A continuación, se presenta una tabla con la definición de algunos conceptos y palabras requeridas para obtener un entendimiento claro del documento.

Acrónimos/Abreviaturas	**Descripción**
API	Application Programming Interface, es un conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

IDE	Integrated Development Environment: es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.
JRE	Java Runtime Environment: Es la implementación de la Máquina virtual de Java que realmente ejecuta los programas de Java. Incluye el JVM, bibliotecas principales y otros componentes adicionales para ejecutar aplicaciones y applets escritos en Java. (Oracle, s.f.)
JDK	Java Development Kit: Se trata de un paquete de software que puede utilizar para desarrollar aplicaciones basadas en Java. Incluye el JRE, conjunto de clases de API, compilador Java, Webstart y archivos adicionales necesarios para escribir applets y aplicaciones de Java. (Oracle, s.f.)
Java SE	Java Platform Standard Edition: proporciona la funcionalidad principal del lenguaje de programación Java. Define todo, desde los tipos y objetos básicos del lenguaje de programación Java hasta las clases de alto nivel que se utilizan para redes, seguridad, acceso a bases de datos, desarrollo de interfaz gráfica de usuario (GUI) y análisis de XML. (Oracle, n.d.)
Java EE	Java Enterprise Edition: está construida sobre la plataforma Java SE. La plataforma Java EE proporciona una API y un entorno de tiempo de ejecución para desarrollar y ejecutar aplicaciones de red a gran escala, de múltiples niveles, escalables, confiables y seguras. (Oracle, n.d.)

JVM	Java Virtual Machine: es una máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el bytecode Java), el cual es generado por el compilador del lenguaje Java. (Wikipedia, 2020)
Servidor de aplicaciones	El servidor de aplicaciones permite el despliegue de aplicaciones empresariales desarrolladas bajo el soporte de las especificaciones Java EE, como por ejemplo Glassfish, WildFly, JBoss.
Servidor WEB	Un servidor web permite la publicación de páginas y servicios web, como por ejemplo Tomcat, Nginx, Internet InformationServer(ISS).
UML	Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

REPRESENTACION DE LA ARQUITECTURA.

La presente solución pertenece al conjunto de servicios establecidos para cumplir las funcionalidades de la aplicación Web Vial Plus, siguiendo un patrón de arquitectura basada en microservicios.

El sistema este compuesto de los siguientes módulos:

Modulo Home: Contiene la interfaz de usuario principal encargada de invocar y realizar las tareas de montaje y desmontaje de las interfaces de usuario de los módulos registrados en el sistema según las elecciones que el usuario realice en el menú de la aplicación; dicho menú se arma dinámicamente acorde a los permisos de acceso asignados al perfil de usuario que ingresa. También presenta la ventana para autenticación de los usuarios.

La siguiente función del este módulo es exponer el API de Módulos con el cual se realiza la instalación, actualización y desinstalación de módulos en la plataforma, adicional a prestar los servicios de descubrimiento, registro y monitoreo de los módulos, sirviendo como visualizador del estado de estos.

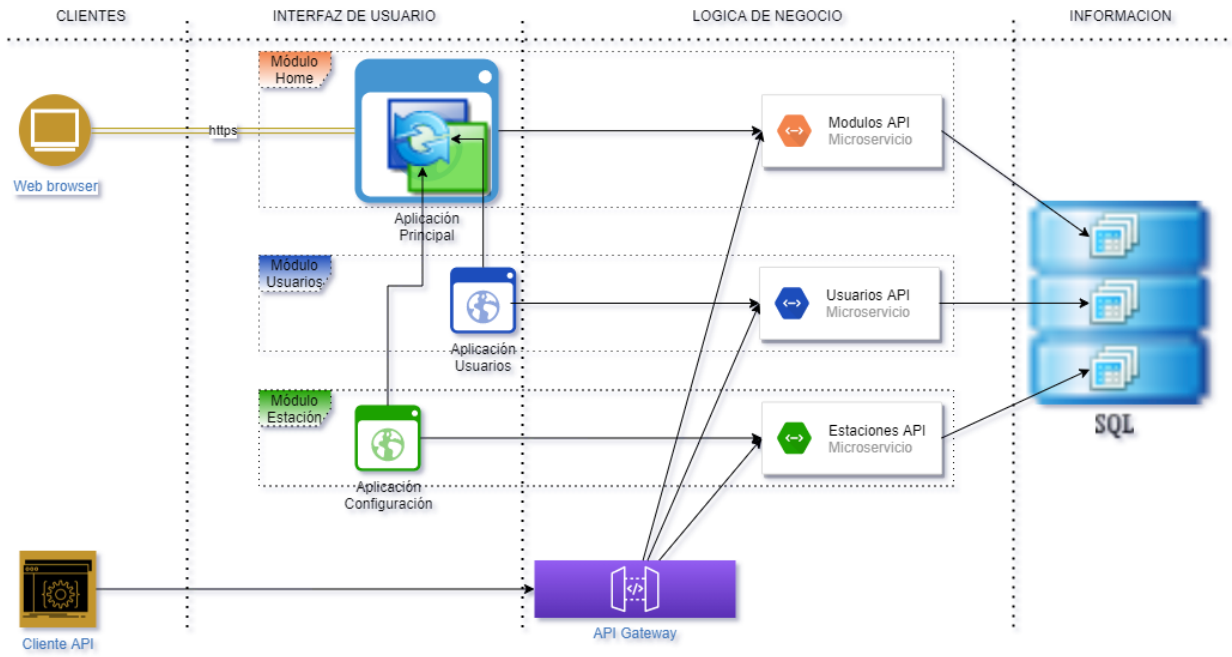
Modulo Usuarios: Contiene las interfaces de usuario y expone los recursos de la API Rest, relacionadas con la administración de usuarios de la plataforma, incluyendo la administración de perfiles y permisos de acceso.

Modulo Estación: Contiene las interfaces de usuario y expone los recursos de la API Rest relacionadas con la configuración de estaciones de peaje, teniendo en cuenta desde información general correspondiente a la Concesionaria Vial y luego aspectos más específicos como lo son los carriles, categorías vehiculares y tarifas de peaje.

En la siguiente imagen se presenta la estructura de alto nivel de la aplicación, en la cual se permite visualizar las interacciones de los diferentes componentes pertenecientes a la plataforma y como los usuarios finales acceden a los servicios prestados:

Figura 49

Diagrama de Arquitectura de Alto Nivel



Nota. Fuente: elaboración propia.

La estructura anterior permite realizar la instalación de nuevos módulos, los cuales deben seguir las siguientes condiciones para la integración a la plataforma:

Las interfaces de usuario deben estar desarrolladas con frameworks basadas en Javascript, debido a que la función de gestión de aplicaciones de la aplicación principal se encuentra basada en funciones de este lenguaje.

Un módulo estándar se compone de la exposición de un microservicio y contener las interfaces visuales para las funcionalidades que vaya a prestar el módulo. Sin embargo, es posible que un módulo se componga solo del microservicio (prestando solamente funciones

de lógica de negocio), pero, no es posible instalar un módulo que solo contenga interfaces de usuario.

La plataforma se compone de los siguientes módulos organizados por dominios. Por convención cada dominio, módulo y opción tendrá un código de identificación que permitirá la gestión de accesos y licenciamiento de la aplicación.

Tabla 7

Opciones por Módulo e Identificadores de Opciones

Opción	Descripción	Código
	Plataforma	**02**
	Módulos	**01**
Consultar		01
Instalar		02
Actualizar		04
Desinstalar		08
	Administración	01
	Usuarios	01
Buscar		01
Registrar		02
Actualizar		04
Enrolar		08
Autenticación		10
Consultar asistencia		20
Registrar asistencia		40

Opción	Descripción	Código
	Perfiles	02
Buscar		01
Crear		02
Modificar		04
Asignar permisos		08
Revocar permisos		10
	Configuración	03
	Concesiones	01
Consultar		01
Crear		02
Modificar		04
	Estaciones	02
Consultar		01
Crear		02
Modificar		04
	Carriles	03
Consultar		01
Crear		02
Modificar		04
	Categorías	04
Consultar		01
Crear		02
Modificar		04
	Tarifas	05
Consultar		01
Crear		02
Modificar		04

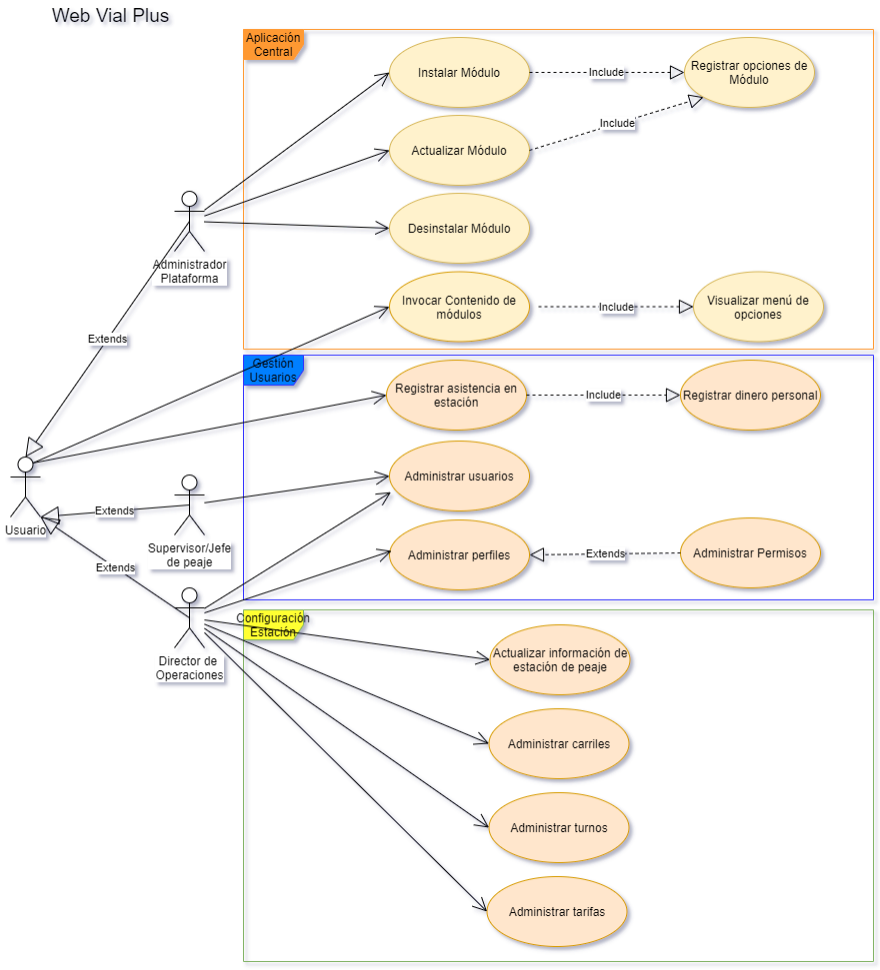
Nota. Fuente: elaboración propia.

DESCRIPCION DE LA ARQUIECTURA.

En la siguiente sección se establecen y detallan las vistas de para los casos de uso significativos para el módulo de Gestión de usuarios.

Vista de escenarios.

Contiene los casos de uso con los cuales se identifican las funcionalidades que cumple el servicio.



Vista lógica.

Entidades.

A continuación, se relacionan las tablas de base de datos establecidas para la administración de los módulos y usuarios:

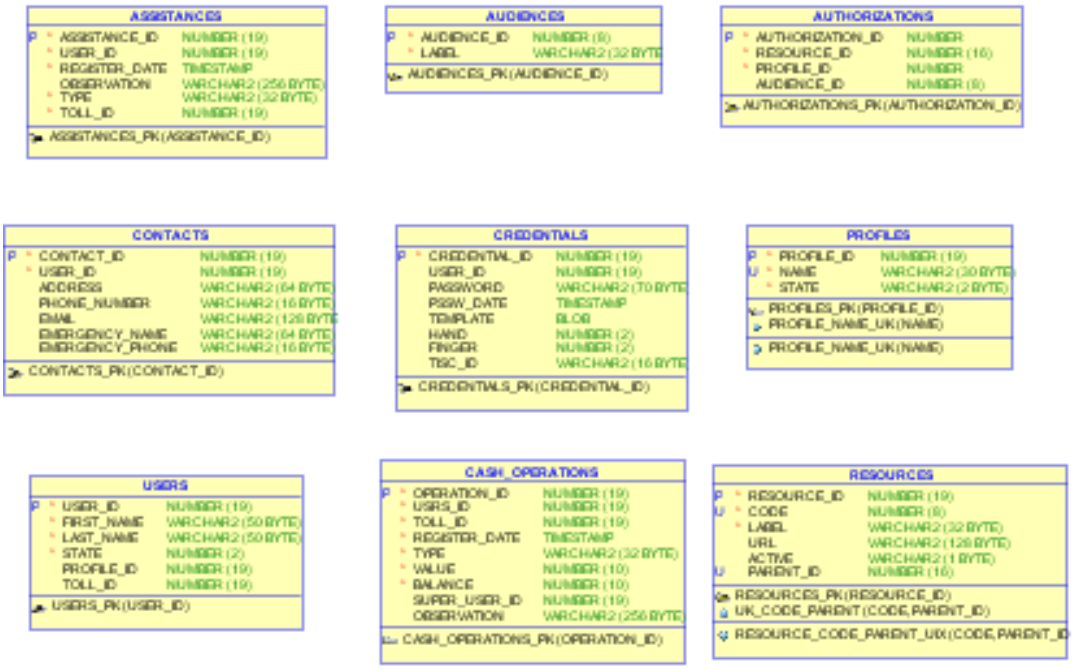


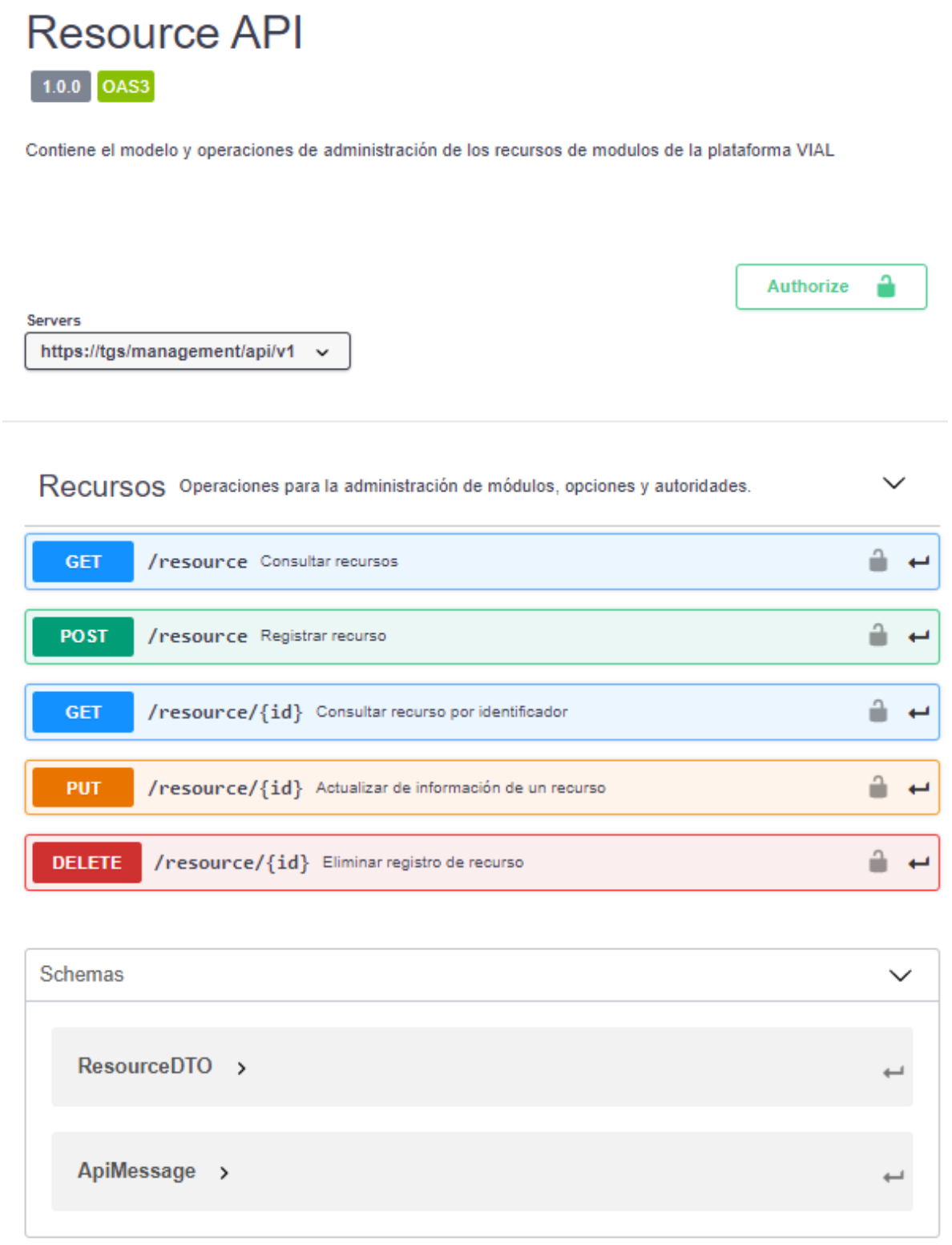
Figura Nº 1. *Entidades - Administración de módulos y usuarios*

Vista de desarrollo.

A nivel de desarrollo se encuentra la definición del API Rest que se establecerán para la interacción de los diferentes módulos.

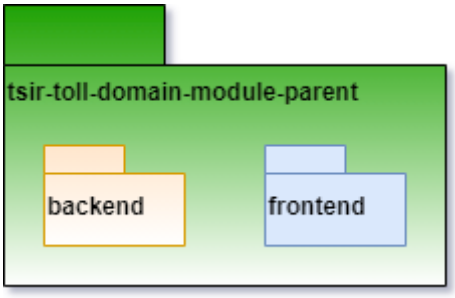
API Rest.

Los módulos son registrados con la siguiente especificación del API Rest, la cual debe estar en la capacidad de gestionar sus contenidos (peticiones y respuestas) en XML y JSON.

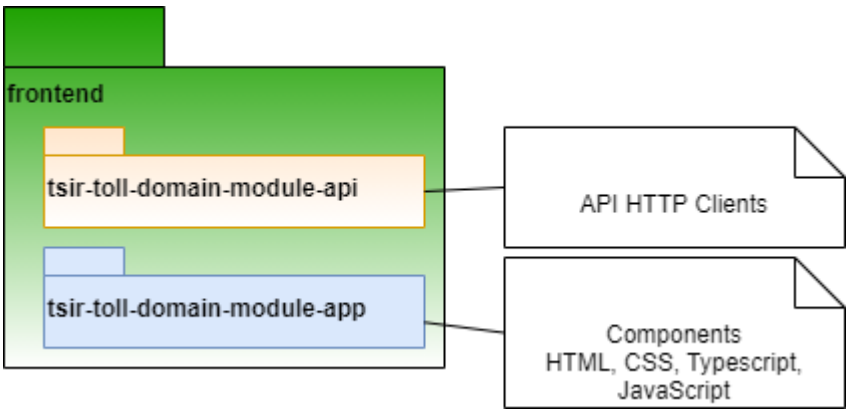


Componentes.

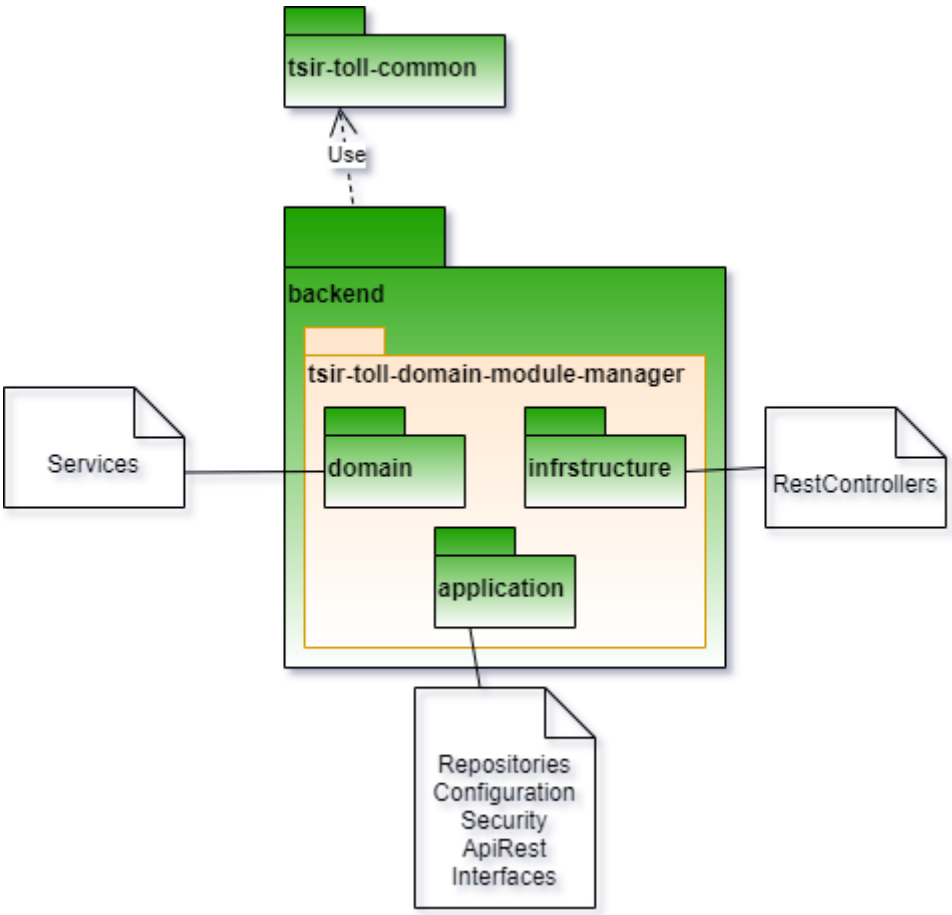
A nivel general la aplicación del microservicio se encuentra dividida en dos componentes principales, articulados como módulos mediante un proyecto padre Maven tipo *POM:*



Componente FrontEnd: en este componente se encuentra la parte de la interfaz gráfica de usuario en el proyecto *tsir-toll-mgmt-usrs-app* que contiene los diferentes componentes de la interfaz gráfica (vistas estilos). Realiza la comunicación con la lógica de negocio siguiendo el API REST mediante una librería generada por el proyecto *tsir-toll-mgmt-usrs-api-ui* (contiene las implementaciones de los clientes HTTP de comunicación):

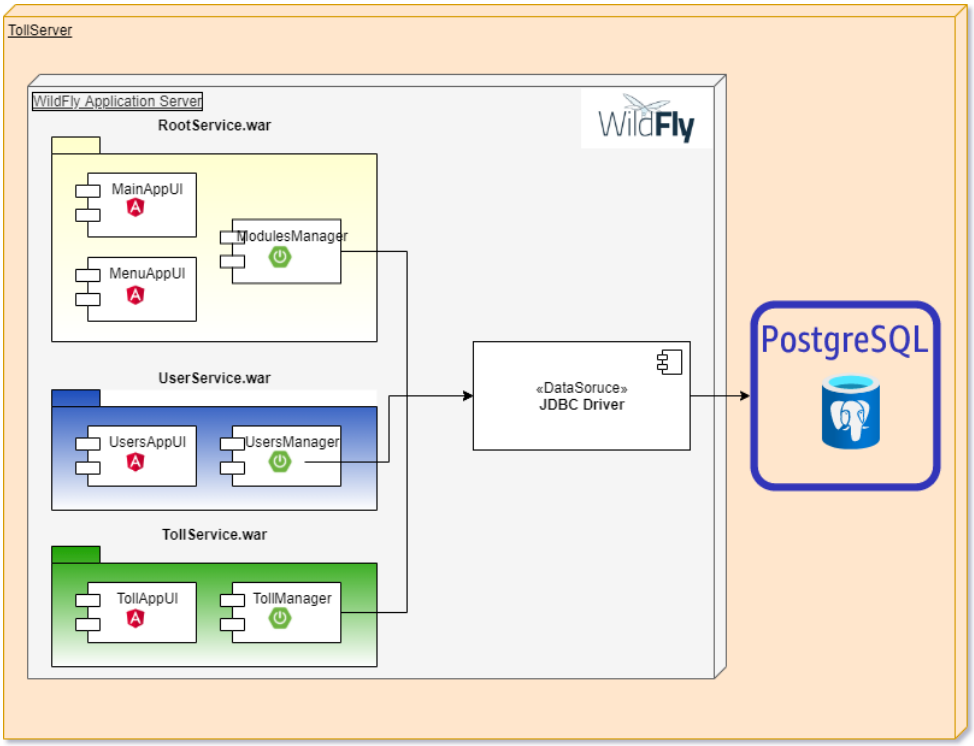


Componente BackEnd: en este proyecto se encuentra la implementación de la lógica del negocio. Corresponde a un proyecto Spring. Utiliza como dependencias el paquete de utilidades *tsir-toll-common*:



Vista física.

El servicio generado corresponde a un archivo WAR que es desplegado en un servidor de aplicaciones (WildFly).

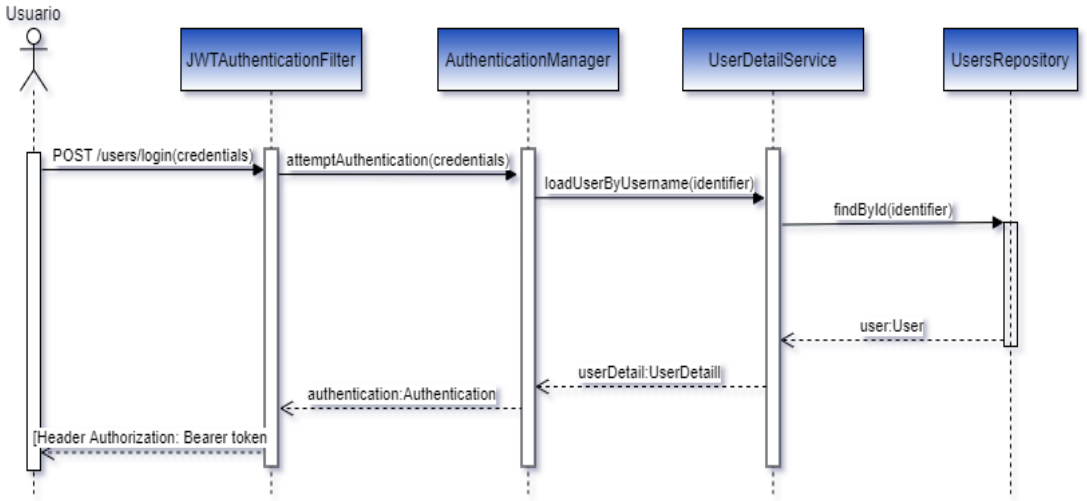


Vista de procesos.

Por una parte, la autenticación de usuarios se realiza tanto para usuario con funcionalidades y operaciones con interfaz de usuario como también para usuarios que consumen las API. El token es generado si la autenticación es correcta y contiene la identificación del usuario y los privilegios asignados al perfil del usuario al que pertenece, adicional a información de vigencia de este. El proceso se presenta el siguiente diagrama de forma general; la funcionalidad es prestada por el módulo de usuarios:

Figura 50

Secuencia para Autenticación de Usuarios

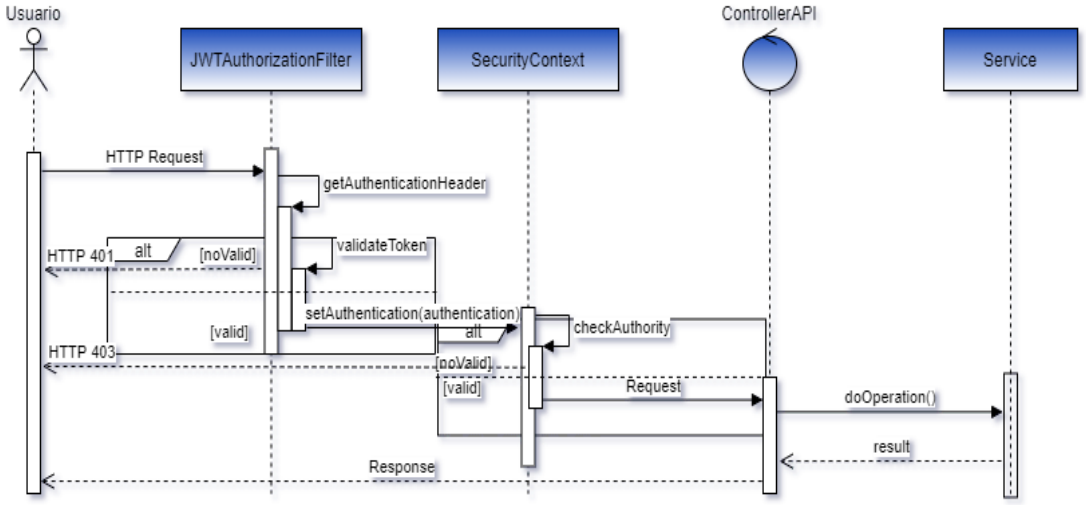


Nota. Fuente: elaboración propia.

Como siguiente proceso, se establece la autorización de los usuarios al acceder a recursos u operaciones de la plataforma. El token generado por el proceso anterior de autenticación debe ser enviado por el cliente en cada solicitud. Cada módulo debe poder verificar su validez localmente, y luego de ello extrae la información necesaria para identificación del usuario y validación de privilegios. En el siguiente diagrama se observa el este proceso con una petición genérica, la cual deben seguir todas las operaciones de los módulos implementados en el sistema:

Figura 51

Secuencia para Autorización de acceso a Recursos

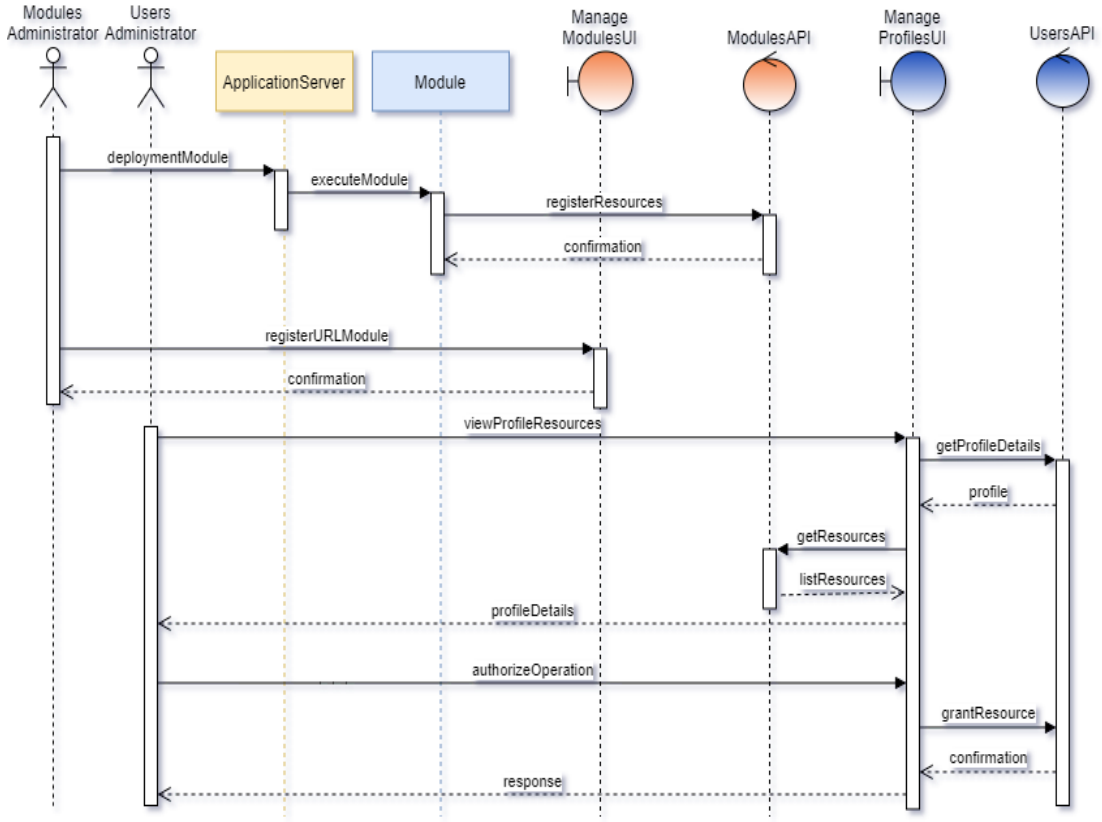


Nota. Fuente: elaboración propia

A continuación, se presenta el proceso de instalación de módulos, mediante el cual es posible además de adicionar nuevos módulos a la plataforma, gestionar los privilegios de acceso a las funcionalidades pertenecientes a ese nuevo módulo:

Figura 52

Secuencia de Instalación de Módulos



Nota. Fuente: elaboración propia

CONSIDERACIONES ADICIONALES.

A continuación, se presentan los factores para tener en cuenta la unificación de criterios en el desarrollo de los diferentes de los módulos de la aplicación.

Requisitos no funcionales.

A continuación, se establecen los requisitos no funcionales, es decir, aquellas características de calidad y restricciones que debe garantizar la aplicación.

Aspecto	Requerimiento
Rendimiento	- El sistema permitirá conexiones concurrentes a la aplicación web, mínimo 20. - El tiempo para carga de la aplicación cuando se acceda a la página principal no será mayor a 5 segundos, y el tiempo de navegabilidad entre pantallas será de máximo 5 segundos. - Se deberá optimizar las imágenes y cantidad de datos que viajan entre el cliente y el servidor.
Robustez	- El sistema contará con un sistema de manejo de errores frente a eventos no planificados, para lo cual se utilizará Log4J para registrar los errores. - Para la gestión de datos y para asegurar la correcta actualización de los datos se utilizará el framework Hibernate, que permite al programador abstraerse del manejo transaccional y centrarse pura y exclusivamente en las operaciones de lógica de negocios.
Seguridad	- El sistema contará con un módulo de seguridad, encargado de la autenticación de usuarios y autorización de las operaciones. La autenticación se realiza mediante contraseña, y tras de ser autenticado se genera un token con el método JWT, en el cual debe contener toda la información necesaria para la autorización de posteriores operaciones en el sistema, teniendo la capacidad de ser validado localmente por cada microservicio. - Los usuarios deberán estar registrados y autenticados. Solo un usuario autenticado podrá realizar las operaciones a las que tenga garantizado el acceso el grupo de usuario asignado. - Las contraseñas deberán cumplir un nivel de complejidad tener 8 caracteres como mínimo y usar mayúsculas y minúsculas. - Las contraseñas debe ser encriptadas antes de ser almacenadas, para eso se utilizará el algoritmo Bcrypt. - Los datos sensibles, tales como plantillas de huelles dactilares, deben ser encriptadas antes de ser almacenadas, para eso se utilizará el algoritmo 3DES. - Las conexiones de comunicación deben ser seguras utilizando HTTPS con soporte de certificados con protocolo TLS 1.2 o posterior.
Escalabilidad	- El diseño del sistema y su construcción deberá contemplar la división entre los datos y la lógica de la aplicación para optimizar la escalabilidad de la aplicación. - Se utilizará el framework Hibernate para mantener independiente el motor de base de datos y su ubicación.

Aspecto	Requerimiento
Diseño	- Esta estructura permite no sólo escalabilidad y reusabilidad, sino también fácil detección de errores además constituye una buena práctica para el desarrollo de aplicaciones empresariales. - El sistema permitirá cambiar su estilo a través de hojas de estilo en cascada (CSS), que deberán personalizarse para cada componente, añadir nuevos estilos y extender componentes para la reutilización de funcionalidad. - Se utilizará las ventajas de la especificación de HTML5. - No se modificará el código de los frameworks para que la aplicación soporte actualizaciones a nuevas versiones. - La aplicación debe tener un diseño Responsive con el fin de poder garantizar la correcta visualización del contenido en distintas resoluciones de pantalla en dispositivos.
Usabilidad	- Los mensajes de error deben ser claros para el usuario final y no deben contener información técnica o detalles de los procesos ejecutados. - La presentación de mensajes para validación deben ser concretos y visualizados debajo del campo validado en color rojo.

Herramientas de desarrollo.

Aplicaciones y versiones requeridas para el ambiente de desarrollo.

Aspecto	Software	Versión	Observaciones
Sistema Operativo	Windows	8.1 o superior	N/A
Servidor DE aplicaciones	WildFly	18.0.1	Sitio oficial: https://wildfly.org/downloads/
IDE	Eclipse IDE	2020-03	Sitio oficial: https://www.eclipse.org/downloads/packages/ Paquete recomendado: Eclipse IDE for Enterprise Java Developers
	Visual Studio Code	1.43.2 o superior	N/A
Java	Java SE Development Kit	8u161 o superior	N/A
	Apache Maven	6.3.X	Sitio oficial: https://maven.apache.org/download.cgi
Angular	NPM Node.js Angular CLI	6.13.4 v12.14.1 8.3.22	N/A
Control de versiones	SVN	N.A.	La ubicación para el almacenamiento del código es provista por la compañía, en servidores encontrados internamente.
Base de datos	PostgreSQL	12.4	N/A

Guías de desarrollo

En la presente sección se establecen las guías que se deben seguir en la codificación de los proyectos, con el fin de establecer un estándar que permita la homogeneidad de los proyectos.

Debido a las características de las aplicaciones desarrolladas para la plataforma se tendrán en cuenta tres aspectos: codificación Java, codificación angular y codificación de base de datos.

ESTÁNDAR DE CODIFICACIÓN JAVA

Para los proyectos de microservicios desarrollados en lenguaje Java tendrán los siguientes lineamientos para la codificación:

Nomenclatura

- El idioma por defecto a la hora de dar sentido funcional al nombre de clases, variables, constantes, etc. será una en inglés y la nomenclatura funcional adoptada. Resumiendo, aquella codificación que por estandarización y/o aceptación se pueda escribir en inglés se mantendrá así por convenio, casos como insert, update, delete, create, retrieve, list, set, get, newInstance, Delegate.
- Para la parte funcional se utilizará inglés, por lo tanto, la nomenclatura de los métodos será: getListCompany en sustitución de getListEmpresa o insertBank en lugar de insertarBanco.

Paquetes

- Por defecto todos los paquetes se escribirán en minúsculas y sin utilizar caracteres especiales. El paquete base queda definido como *org.tsir.toll*, en este paquete no se definirá ninguna clase.
- Se tendrá, así mismo, niveles extra dentro del paquete definido, como el nombre del dominio y del módulo, es decir, *org.tsir.toll.<domain>.<module>*.
- El proyecto que contiene componentes comunes a varios de los módulos implementados, el nombre de los paquetes comenzarán por: *org.tsir.common*
- La estructura en árbol de los paquetes siguientes a partir del paquete base se define como sigue:
    - application
        - presentation
        - register
        - utils
    - domain
        - dto
        - entities
        - services
    - infrastructure
        - api
        - config
        - persistence
        - security

Nombres de Interfaces

- Los nombres de interfaces terminarán en el sufijo *able* indicando funcionalidades implementables y estarán compuestos por palabras con la primera letra en mayúscula (CamelCase). Se debe evitar el uso de abreviaciones que dificulten la comprensión del código. Ejemplo: <<Registrable>>

Nombres de clases

- Los nombres de clases deben ser mezclas de mayúsculas y minúsculas, con la primera letra de cada palabra interna en mayúsculas (CamelCase).
- Debemos intentar mantener los nombres de clases simples y descriptivos.
- Debemos usar palabras completas y evitar acrónimos y abreviaturas (se permiten DAO, DTO, URL, HTML, etc.). Si la clase cumpliese algún patrón determinado o tuviese una funcionalidad específica es recomendable definirlo en el nombre.

Paquete	Funcionalidad	Nombre
bussines.dao	Data Access Object (Interface)	<nombre>DAO
bussines.dao.impl	Data Access Object (Implementation)	<nombre>DAOImpl
bussines.exception	Excepciones	<nombre>Exception
bussines.service	Service	<nombre>Service
bussines.helper	Helper	<nombre>Helper
bussines.dto	Data Transfer Objects	<nombre>DTO
util	Clases de constantes.	<scope>Keys <nombre>Keys
web.controller	Controller	<nombre>Controller
web.filter	Filter	<nombre>Filter
web.model	Model	<nombre>Model
web.listener	Listener	<nombre>Listener

Nombres específicos de gestiones

- Cuando se trata de gestionar una entidad determinada (Ej. User) se definen los nombres de clases, demás ficheros implicados con las siguientes reglas:

  Clase: <<FuncionalidadGenerica>><<Entidad>><<Especificación de Clase>>

  Ejemplo: User, UserAction, FindUserAction

Métodos

- Los métodos deberán ser verbos (en infinitivo), en mayúsculas y minúsculas con la primera letra del nombre en minúsculas, y con la primera letra de cada palabra interna en mayúsculas (lowerCamelCase).
- No se permiten caracteres especiales.
- El nombre ha de ser lo suficientemente descriptivo, no importando la longitud de este.

Variables

- Los nombres de las variables tanto de instancia como estáticas reciben el mismo tratamiento que para los métodos, con la salvedad de que aquí sí importa más la relación entre la regla mnemónica y la longitud del nombre.

  Correctos: isActive, registerDate

  Incorrectos: dC, DCal, fI, FI

- Se evitará en la medida de lo posible la utilización de caracteres especiales, así como nombre sin ningún tipo de significado funcional.
- Las excepciones son las variables utilizadas en bucles for, para esos casos se permite utilizar i, j, k, l y siempre en ese orden de anidamiento.
- El primer bucle siempre será el que tenga la variable i como iterador. (Esta variable se definirá para el bucle en cuestión).

Constantes

- Los nombres de constantes de clases deberían escribirse todo en mayúsculas con las palabras separadas por guion bajo ("_"). Todas serán declaradas como public static final. Ej.

```
public static final String PROPERTY_URL_SERVICE =
"https://urlService";
```

Comentarios

- Los comentarios serán utilizados para dar información adicional al desarrollador sobre la implementación del diseño de la clase. Se tiene, por tanto, que evitar referencias al diseño funcional de la misma. El uso abusivo de los comentarios es desaconsejable, principalmente por el trabajo extra necesario para su correcto mantenimiento. Es preferible rediseñar el código para una mejor compresión de este.
- Se tienen que evitar el uso de caracteres especiales dentro de los comentarios, así como el uso de cajas u otro tipo de gráfico creado mediante códigos ASCII.
- La estructura de los diferentes tipos de comentarios y su uso general se presenta en la clase base de codificación.

Declaraciones

- Para la declaración de las variables se utiliza una declaración de cada vez y no se permiten dejar variables locales sin inicializar salvo en el caso de que sean propiedades de un objeto bean. La codificación correcta sería:

```
public static Integer entero = new Integer(0);
```

- La declaración de las variables locales a una clase, método o bloque de código se realizan al principio de este y no justo antes de necesitarse la utilización de la variable.
- La única excepción a esta regla son las variables que gestionan los bucles for.

- Las variables de avance de bucles for no podrán ser modificadas de ninguna manera fuera de la propia sentencia del bucle.
- La duplicidad de los nombres de variables en diferentes niveles dentro de la misma clase se tiene que evitar.

Sentencias

Las normas básicas son:

- Una sentencia por cada línea de código.
- Todo bloque de sentencias entre llaves, aunque sea una sola sentencia después de un if.

Buenas prácticas

Constantes

- Como norma general todas las constantes numéricas no deberían codificarse directamente, salvo la excepción de -1, 0 y 1.

Propiedades

- El acceso/modificación de las propiedades de una clase (no constantes) siempre mediante métodos de acceso get/set.
- La asignación de variables / propiedades no podrá ser consecutiva.

```
Variable1 = variable2 = "hola mundo"; //No válido
```

- No utilizar el operador asignación en sitios donde se pueda confundir con el operador igualdad. Ni dentro de expresiones complejas.

Métodos

- Como norma general no se debe acceder a un método estático desde una instancia de una clase, debemos utilizar la clase en sí misma.
- Es altamente recomendable utilizar plugins que permitan el análisis estático del código. Por ejemplo, para Eclipse IDE y VS Code se encuentra el plugin SonarLint.

CONVENCIONES SQL

En esta sección se establecen los lineamientos a seguir en la definición de la estructura de bases de datos.

Recomendaciones generales

- Solo utiliza caracteres del idioma inglés o ASCII para objetos de la base de datos como el nombre propio del esquema, tablas, campos, índices, llaves foráneas, usuarios, procedimientos almacenados, etc…
- Se utilizarán el idioma inglés para la definición de objetos y mayúsculas.

Base de datos

- El charset de las bases de datos debe ser ‘UTF-8′.

MSSQL Server=‘SQL_Latin1_General_CP1_CI_AI’
Oracle=’AL32UTF8’ o ’AL32UTF16’
Postgres = ‘UTF-8’

Tablas

- El nombre de las tablas debe ser en mayúsculas y plural, deben ser descriptivos, no importa que tan largos sean siempre y cuando sean soportados por la base de datos. Ej. 'USERS'.
- Si la tabla tiene más de 2 palabras estas se deben separar con un guion bajo, nunca se debe de usar espacios, ej. 'USER_CONTACTS'.
- Si la tabla es una tabla muchos a muchos (multivaluada), se deben de utilizar los nombres de las tablas que generan la relación, deberán de ir con guiones bajos, ej 'SELLER_CLIENT', 'SELLER_CLIENT_PRODUCT'.
- Evitar el uso de tablas temporales, utilizar estructuras de datos del lenguaje de programación que se esté usando.
- Si se requiere guardar el histórico de una tabla en otra tabla diferente, utiliza el prefijo HIST, donde los campos de la tabla histórica son prácticamente los mismos que los de la tabla incluyendo el ID de la tabla original que en este caso quedará como un campo normal, así mismo se le agrega un marcado de tiempo, el cual puede ser una simple fecha o una estampa de tiempo. También hay que evaluar los índices para esta tabla ya que muy probablemente tengan que ser diferentes a los de la tabla original.

Campos

- El nombre de los campos/columnas deben ser en singular.
- Todas las tablas deben de tener una llave primaria (PK) y debe de ser el primer campo de la tabla y debe de ser único e irrepetible. El nombre del PK debe ser el nombre de la tabla en singular terminado en guion bajo y el sufijo ID, ej. 'TABLE_NAME_ID'.
- El nombre de los campos debe de ir en mayúscula usando un guion bajo como espacio.

- Evitar el uso de campos en nulo, todos los campos deben de tener una inicialización, esta puede ser vacía y en caso de que tenga otra, debe de estar comentada en la tabla, a menos que se requiera determinar si tiene o no un valor.
- Indexar campos de fechas.
- Al crear los campos siempre se debe de dejar una holgura o margen de maniobra, ya que es común que los requerimientos evolucionen o cambien puedan provocar generar problema al momento que estos afecten la base de datos.

Transacciones

- El uso de transacciones es obligatorio cuando se inserta, actualiza o se elimina más de una tabla.
- Trata de usar niveles de bloqueo (isolation level) optimistas para reducir el tiempo de bloqueo de la tabla, puede ser ‘Read uncommitted’ o ‘Read committed’.

Relaciones

- Todas las FK deben de tener relación con restricciones de updates en cascada y borrado restringido.
- Todas las llaves foráneas deben de ser del mismo tipo de dato y longitud que la llave primaria a la que hace referencia.
- Todas las llaves foráneas deben de ir indexadas en las tablas donde son foráneas, de preferencia con índices descendentes.
- Se deberá de evitar la redundancia cíclica, esto sucede cuando se tiene una actualización en cascada desde 2 o más tablas padres a un campo de la tabla hija, en este caso solo se debe de mantener una sola actualización en cascada o su defecto ninguno.

Procedimientos almacenados

- Eliminar el uso de procedimientos.

Funciones

- Eliminar el uso de funciones (las desarrolladas por el usuario).

Triggers (gatillos o disparadores)

- Eliminar el uso de Triggers.

Vistas

- Evitar (tratar de no usar) el uso de vistas.

Seguridad

- Para conectar un sistema, crea un "usuario funcional" que solo tenga privilegios de SELECT, INSERT, UPDATE, DELETE, SESSION para un esquema que va a utilizar, QUE NO TENGA darle privilegios de DROP y/o GRANT.
- Limitar el privilegio al mínimo necesario para acceso y funciones a los usuarios de base de datos.
- Llevar un control de versiones de la base de datos.
- Para todo cambio en la base, también se debe de actualizar el diagrama de la base de datos.
- Bloquear el acceso a la base de datos a través del firewall.
- Escanear periódicamente la red.
- Limitar el acceso físico y lógico a la red y al servidor de base de datos.
- Evitar realizar tareas de administración remotas, si no es posible evitarlas utiliza una VPN.

- Nunca exponer una base de datos a Internet.
- Deshabilitar el usuario súper administrador, sa, root o toor.
- Cambiar las contraseñas que vienen por defecto.
- Habilitar la encriptación (SSL) de las bases de datos y de la conexión. O en su defecto usar un túnel SSH.
- Para almacenar las contraseñas de los usuarios utiliza métodos de hasheo a nivel aplicación (vía lenguaje de programación) en vez de los algoritmos que proporciona la base de datos.
- Revisar periódicamente el espacio en disco y los logs de la base.
- Usa herramientas de monitoreo de consultas (revisar deadlocks) y log de conexiones a la base de datos.
- Antes de realizar un cambio importante a la base de datos, realizar una copia de seguridad.
- Realizar respaldos periódicamente. Programar copias de seguridad automáticas para que se ejecuten cada determinado tiempo.
- Todas las contraseñas deberán de ser almacenados vía hash superior o igual a RSA-512, no usar ningún algoritmo menor ej. (MD5 ó SHA-256), ya son obsoletos e inseguros.
- Si se va a almacenar datos sensibles como puede ser un número de tarjeta se deberá de almacenar de manera encriptada y deberá de ser desencriptada por la aplicación.
- Aplicar los parches de seguridad la base de datos por lo menos una vez cada 6 meses.
- Limitar los accesos al sistema operativo y servidor en donde esta alojada la base de datos.
- Actualizar periódicamente el Sistema Operativo en donde esta alojada la base de datos.

ESTÁNDAR DE CODIFICACIÓN ANGULAR

Para esta sección se establecerán los lineamientos a seguir en los recursos de los proyectos Angular en el cual se va a realizar el desarrollo de la parte frontend (interfaces de usuario), es decir, archivos TypeScripts, JavaScript, Html y CSS, adicionalmente de la estructura recomendada de los proyectos.

Planteamiento metodología de seguridad

Security Development Lifecycle (Microsoft)		
Etapa	Fase	Actividades
Formación	Formación	Lectura normativas y marcos de trabajo de seguridad en las aplicaciones. (OWASP, Spring Security, JSON Web Token)
	Requisitos	Identificar modelo de control de acceso. (controles de perfilamiento de usuarios). Autenticación, autorización
		Identificar datos sensibles para encriptar. (contraseñas, huellas). (almacenamiento aes)
		Identificar los riesgos de seguridad del negocio.
		Evaluar información de privacidad de los usuarios.
		Diseñar el modelo de control de acceso de la aplicación.
		Plan de integración con el área de seguridad de la compañía.
		Diseñar los medios de autenticación para usuarios de la aplicación.
		Establecer métodos y algoritmos de encriptación para datos sensibles.
		Diseñar el modelo de seguridad de las API Rest.
		Planteamiento de componentes: Spring Security, JSON Web Token.
	Diseño	Diseño de pruebas de seguridad.
Mejora continua		Evaluación de los controles para tipos de ataque conocidos. Cross Site Scripting, SQL Injection.
		Diseño de filtros de control y sus excepciones. Cors (cross-origin resource sharing), frame access.
		Establecimiento de los métodos de seguridad para la capa de transporte y conexión. SSL bajo TLS v1.2.
		Plan de manejo de contraseña para bases de datos, servidor de aplicaciones.
		Evaluar riesgos de repositorios de almacenamiento de código establecidos por la compañía para las fuentes de código.
	Implementación	Evaluar los riesgos y vulnerabilidades de las librerías externas.
		Implementar técnicas de ofuscación de código para recursos visibles (HTML, CSS, JavaScript).
		Ejecutar pruebas de acceso.
	Comprobación	Ejecutar pruebas de autenticación.
		Ejecutar pruebas de transporte.
	Lanzamiento	Establecimiento de los métodos para identificación, manejo y escalamiento de los incidentes de seguridad identificados en producción.
Responsabilidad	Respuesta	Establecer plan de respuesta a incidentes de seguridad.

Aspectos de Gobierno estratégico

1. Partes interesadas.
    a. Internos.
        - Gerencia Operativa.
        - Área de Análisis y Diseño.
        - Área de Desarrollo.
        - Área de QA.
        - Área de Soporte.
        - Área de Infraestructura y redes.
        - Área de seguridad.
        - Gerencia comercial.
    b. Externos.
        - Concesiones (Clientes).
        - Proveedores.
        - Universidad.

2. Listas de beneficios, riesgos y recursos.

Ámbito	Aspectos
Beneficios	• Reducción los tiempos de entrega del software gracias a la división de módulos desarrollados y desplegados independientemente. • Incremento de la calidad del software producido por la compañía, estableciendo lineamientos para el uso de patrones, estándares y procesos definidos de desarrollo. • Establecimiento de una arquitectura de software como herramienta que sirva como base para el desarrollo de nuevos módulos y migración de actuales funcionalidades en una plataforma nueva y optimizada en comparación con la plataforma existente.

	• Eliminar la necesidad actual de un despliegue completo requerido en las actualizaciones de la plataforma actual.
Riesgos	• Dificultad en la compatibilidad con plataforma anterior para garantizar la continuidad de los procesos del cliente. • Periodo de estabilización prolongado para la nueva plataforma.

Recursos	• Licencias de Sistema Operativo, herramientas ofimáticas y herramientas de seguridad (Antivirus cortafuegos).(Proveído Organización).
	• Tiempo:
	  o 40 horas/semana.
	  o 30 semanas.
	• Hora/Desarrollo:
	  o $ 25.000 COP.
	• Herramientas de desarrollo libres/gratuitas.
	• Recursos de infraestructura (servidores y repositorios de código privados en sitio) proveídos por la organización. No se conocen costos.

Tabla 1 Lista de beneficios, riesgo y recursos. Fuente: Elaboración propia.

3. Selección de metas por dimensión.

Dimension	Objetivos empresariales	Objetivos relacionados con TI
Financiera		
	1. (P)Valor para las partes interesadas de las inversiones del negocio	
		ORTI 01 Alineamiento de TI y estrategia de negocio
		ORTI 13 Entrega de Programas que proporcionen beneficios a tiempo, dentro del presupuesto y satisfaciendo los requisitos y normas de calidad.
		ORTI 17 Conocimiento, experiencia e iniciativas para la innovación de negocio
	2. (S)Cartera de productos y servicios competitivos	

		ORTI 05 Realización de beneficios del portafolio de Inversiones y Servicios relacionados con las TI
		ORTI 07 Entrega de servicios de TI de acuerdo con los requisitos del negocio
		ORTI 08 Uso adecuado de aplicaciones, información y soluciones tecnológicas
		ORTI 11 Optimización de activos, recursos y capacidades de las TI
Cliente	**6. (P)Cultura de servicio orientada al cliente**	
		ORTI 01 Alineamiento de TI y estrategia de negocio
		ORTI 05 Realización de beneficios del portafolio de Inversiones y Servicios relacionados con las TI
		ORTI 07 Entrega de servicios de TI de acuerdo con los requisitos del negocio
		ORTI 13 Entrega de Programas que proporcionen beneficios a tiempo, dentro del presupuesto y satisfaciendo los requisitos y normas de calidad.
	10. (S)Optimización de costos de entrega de servicio	
		ORTI 08 Uso adecuado de aplicaciones, información y soluciones tecnológicas
		ORTI 11 Optimización de activos, recursos y capacidades de las TI
		ORTI 12 Capacitación y soporte de procesos de negocio integrando aplicaciones y tecnología en procesos de negocio
Interno	**12. (P)Optimización de los costes de los procesos de negocio**	
		ORTI 05 Realización de beneficios del portafolio de Inversiones y Servicios relacionados con las TI
		ORTI 06 Transparencia de los costos, beneficios y riesgos de las TI
		ORTI 11 Optimización de activos, recursos y capacidades de las TI
		ORTI 13 Entrega de Programas que proporcionen beneficios a tiempo, dentro del presupuesto y satisfaciendo los requisitos y normas de calidad.
	13. (S)Programas gestionados de cambio en el negocio	
		ORTI 04 Administración de los riesgos de negocio relacionados con las TI gestionadas
		ORTI 09 Agilidad de las TI
		ORTI 12 Capacitación y soporte de procesos de negocio integrando aplicaciones y tecnología en procesos de negocio
Aprendizaje	**16. (S)Personas preparadas y motivadas**	
		ORTI 08 Uso adecuado de aplicaciones, información y soluciones tecnológicas
		ORTI 16 Mantener las personas preparadas y motivadas
	17. (P)Cultura de innovación de producto y negocio	
		ORTI 01 Alineamiento de TI y estrategia de negocio
		ORTI 05 Realización de beneficios del portafolio de Inversiones y Servicios relacionados con las TI
		ORTI 08 Uso adecuado de aplicaciones, información y soluciones tecnológicas
		ORTI 17 Conocimiento, experiencia e iniciativas para la innovación de negocio.

Tabla 2 Relación de objectivos empresariales y objetivos de TI. Fuente: basado en 2. COBIT5-Assessment-Scoping-Tool_BSApr14.xlsx

4. Elementos para cada uno de los habilitadores de gobierno.

Habilitadores	Elementos
Principios, políticas y Marcos.	• Misión. • Visión • Valores. • Política de calidad. • Política de seguridad de la información. • Resolución 4303 de 2015 "Por la cual se reglamenta la interoperabilidad de peajes con recaudo electrónico vehicular (ip/rev)". • Ley 1581 de 2012 "Régimen General de Protección de Datos Personales"
Procesos.	• Administración de usuarios • Configuración de estaciones de peaje. • Administración de módulos de aplicación.
Cultura, ética y comportamiento.	Se encontraría regido por los valores empresariales.
Información.	• Recaudo de peajes. • Trafico de peajes. • Datos de personal operativo.
Servicios de infraestructura y aplicaciones	• Herramientas de gestión de Incidentes GLPI y Redmine. • Directorio activo. • Servidores de repositorio de código SVN.

Personas, habilidades y competencias	• Empleados relacionados con TI. • Empleados operativos.

Tabla 3 Elementos de los habilitadores de gobierno. Fuente: elaboración propia.

5. Selección procesos genéricos de COBIT.

   Finalmente, de los 36 o 37 procesos genéricos de COBIT seleccionen los más importantes para su proyecto.

Process ID	Procesos para la gobernanza de la tecnología empresarial
	Evaluar dirigir y monitorear
EDM02	Asegurar la entrega de beneficios a las partes interesadas.
EDM03	Asegurar la optimización del riesgo tecnológico
EDM04	Asegurar la optimización de recursos
	Alinear, planear y organizar
APO02	Gestionar la estrategia tecnológica
APO03	Administrar la arquitectura empresarial
APO04	Gestionar la innovación tecnológica
APO05	Administrar portafolio
APO11	Gestionar la calidad tecnológica
APO13	Administrar la seguridad tecnológica
	Construir, adquirir e implementar
BAI01	Gestionar programas y proyectos
BAI04	Gestionar disponibilidad y capacidad
BAI06	Gestionar cambios tecnológicos
BAI07	Gestionar la aceptación de cambios y la transición
BAI10	Administrar la configuración
	Entrega, servicio y soporte

DSS01	Gestionar operaciones
DSS02	Gestionar solicitudes de servicio e incidentes
DSS04	Gestionar la continuidad
	Monitorear, evaluar y verificar
MEA01	Monitorear, evaluar y controlar el desempeño y la conformidad de TI
MEA03	Monitorear, evaluar y controlar el cumplimiento de requisitos externos.

Tabla 4 Selección de procesos genéricos de COBIT. Fuente: basada en (ISACA, 2012)