Sistema de control de acceso real-time

Autores:

- Belcic, Martin
- Luna, Tomas
- Viejo, Ignacio

Director:

- MBA Genin, Fernando

Proyecto final para optar al grado de Ingeniero Informático de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata

Mar del Plata, 17 de agosto de 2023

Sistema de control de acceso real-time

Autores:

- Belcic, Martin
- Luna, Tomas
- Viejo, Ignacio

Director:

- MBA Genin, Fernando

Proyecto final para optar al grado de Ingeniero Informático de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata

Mar del Plata, 17 de agosto de 2023

Agradecimientos

Este proyecto no podría haberse llevado a cabo si no fuera por todos los docentes que enseñan en la carrera de Ingeniería en Informática. Los conocimientos adquiridos fueron fundamentales para todas las etapas. Cada una de las materias cursadas dejó un aprendizaje que hizo posible la resolución del problema.

La labor del director del proyecto, Fernando Genin, también fue fundamental ya que brindó consejos, sugerencias, correcciones y resolvió las dudas en todo momento. Además, fue quien acercó el proyecto al equipo. Sin él esto no hubiera ocurrido.

Por último, los familiares y amigos ofrecieron contención y apoyo incondicional en los momentos más difíciles, sobre todo cuando el objetivo parecía estar tan lejano. Esas fuerzas y energías hicieron que este proyecto no quede en el olvido.

Muchas gracias.

Índice

Resumen

El objetivo principal del proyecto es desarrollar un sistema que permita controlar los accesos de los empleados y contratistas a la empresa Nutreco. También debe contar con la posibilidad de cargar y verificar su documentación. Se busca que el sistema sea fiable, intuitivo y tenga la capacidad de brindar reportes acerca de las personas que se encuentran en la planta en un determinado momento.

Lo primero que se realizó fue un análisis profundo sobre las necesidades y requerimientos del cliente. Se mantuvieron reuniones presenciales y virtuales con el equipo de sistemas para obtener información acerca de las tecnologías con las que contaba la empresa. Se realizaron los diagramas de arquitectura, base de datos, componentes, secuencia, entre otros. Además, el cliente aprobó los bocetos de la página web y la elección de los lenguajes de programación a utilizar junto con sus respectivos *frameworks*.

Se realizó un producto de forma iterativa e incremental utilizando metodologías ágiles. La instalación de una aplicación móvil con lector de código QR en la puerta principal logró un acceso simple y rápido a la planta. A su vez, se realizaron pruebas tanto unitarias como de integración para garantizar su correcto funcionamiento. Se instalaron los diferentes módulos en el servidor de la empresa, se cargaron los datos principales, se realizaron capacitaciones a los usuarios para su correcta utilización y se transfirió el conocimiento al área de sistemas de la organización para su posterior mantenimiento.

Actualmente el sistema se encuentra instalado y en funcionamiento. Es de gran utilidad para la empresa: Aceleró los tiempos de acceso, facilitó el control y administración de la documentación, permitiendo realizar un seguimiento sobre la cantidad de personas dentro de la planta para cumplir con la capacidad máxima.

1. Introducción

Nutreco es una empresa que desarrolla productos de alta calidad y sustentables. Entre sus productos se encuentra el agua Sierra de los Padres y las gaseosas Goliat. Su planta principal se encuentra en la Sierra de los Padres.

La empresa realizaba el seguimiento y control de la documentación y acceso a planta de sus empleados y contratistas de forma manual. Este proceso era lento y poco práctico por lo cual, se lo necesitaba informatizar lo antes posible. Se perdía mucho tiempo, se necesitaba personal dedicado a esta labor y, además, se generaban inconsistencias y errores en forma permanente.

Los documentos se almacenaban en papel y su seguimiento era en planillas completadas por un personal de la empresa. Cada vez que un nuevo conductor se anunciaba en la puerta se debía verificar manualmente en la planilla que todo fuera correcto. En caso de alguna duda o inconveniente se debía buscar el documento físico que estaba almacenado en algún cajón del lugar. Las planillas debían ser mantenidas ya que constantemente se recibía nueva documentación cuando esta se vencía.

En cuanto al ingreso y egreso de la planta, se registraba en una planilla que era completada por el personal de seguridad. Al contar con una gran cantidad de proveedores resultaba muy difícil determinar la cantidad de personas en planta en un momento determinado. Este es un dato importante y fundamental en un supuesto incidente que requiera evacuación de la planta o la limitación en la cantidad de personas circulantes, situación que se vio agravada por la pandemia COVID-19.

2. Objetivo del Proyecto

2.1 Objetivo General

El principal objetivo del proyecto es el desarrollo de un sistema que permita la carga, validación y control de documentación de los empleados y contratistas de la empresa Nutreco SA en forma rápida, fácil y eficiente, lo cual permitirá agilizar tiempos, reducir costos y minimizar errores. Además, debe permitir controlar el acceso a planta, automatizando el ingreso de datos y evitando demoras para los usuarios.

2.2 Alcance inicial

En principio el sistema debía incluir el control de documentación de parte de empleados y contratistas, control del acceso a la planta, servicio de facturación con los proveedores transportistas, y dos redes neuronales: La primera para la detección de anomalías en el horario de ingreso y egreso de los empleados de la empresa y, la segunda, capaz de monitorear y validar los documentos previo a la auditoría del personal de la empresa, de manera que se simplifique el trabajo de los mismos y se evite la carga de documentos incompletos o de mala calidad.

Los datos recopilados permitirán construir estadísticas y avisos útiles tanto para los contratistas como para el personal de Nutreco. Se podrá saber la cantidad de personas en planta, obtener reportes de permanencia personalizables, recibir avisos de documentación vencida o por vencer, entre otros. Además, los datos serán de utilidad para el entrenamiento de las dos redes neuronales.

Por pedido del cliente, el sistema primero se implementó en la planta ubicada en San Rafael para luego extenderlo a la planta de Sierra de los Padres. La plataforma debe comportarse como un único sistema sin importar la cantidad de plantas.

2.3 Alcance final

Durante el transcurso del proyecto se produjeron varios cambios de requerimientos por parte del cliente que derivaron en la modificación del alcance del mismo. Estos cambios y desvíos serán tratados en detalle en los siguientes capítulos. Por lo tanto, el alcance final del sistema fue: control de documentación de parte contratistas, control del acceso a la

planta a través de un personal de expedición ajeno a la empresa, control de documentación y administración de visitas de empleados tercerizados, red neuronal capaz de monitorear y validar los documentos previo a la auditoría,

El objetivo principal del proyecto no se modificó. Es decir, siguió siendo el control de documentación y acceso a planta de empleados y contratistas, con la diferencia que en los empleados de la empresa fueron sustituidos por empleados tercerizados.

3. Proyecto

3.1 Problema a resolver

El control tanto del personal como de los contratistas y de su documentación dentro de la empresa Nutreco es un proceso manual, hecho por un encargado en portería, y poco eficiente. Esto genera inconvenientes a la hora de controlar que la documentación cumpla con la reglamentación, debiendo en ocasiones, permitir el ingreso de contratistas fuera de regla. También, el estado de dicha documentación suele ser desconocido, sin saber si esta fue entregada, está próxima a vencer o está vencida.

El control del ingreso de los empleados se ve afectado por este proceso. El sistema actualmente utilizado para controlar los horarios de ingreso y salida de los mismos es manual. Esto permite que los mismos fichen fuera de horario, o incluso en lugar de otros colegas. Tampoco se tiene información precisa sobre la cantidad de personas dentro la planta en un determinado momento. Este requerimiento finalmente fue reemplazado por pedido de la Empresa, debido a que necesitaban incorporar un seguimiento sobre las visitas de los empleados tercerizados, quienes seguían un flujo diferente al de las demás entidades.

Tampoco cuentan con un mecanismo para poder llevar un registro certero de a quien se les ha permitido ingresar fuera de norma y tampoco poseen un registro sobre las actividades de los contratistas dentro de la planta.

Además, como parte de la solución se encuentra el proceso de facturación con los proveedores transportistas, ya que no se encontraba automatizado. Las facturas, remitos,

ordenes de compra, etc. deben ser intercambiadas personalmente entre las partes involucradas.

3.2 El equipo

La confianza y la amistad es una de las grandes características del equipo. La sinergía entre los integrantes es algo que existía desde las cursadas, trabajos prácticos y proyectos profesionales. Esto hizo que la comunicación sea fácil, el proceso se disfrute más e incluso, se pueda hacer una mejor autocrítica del equipo para desarrollar un mejor producto.

Además, la distribución de conocimiento es algo muy notable en el grupo. Cada integrante tuvo diferentes especializaciones previas que llevaron a que se logre cubrir gran parte de las diferentes áreas que se necesitaban para el proyecto. Los miembros pudieron enfocarse y profundizar sus habilidades en el rol que le tocó cumplir. Algunas de ellas fueron: desarrollo *backend* y *frontend*, arquitectura, base de datos, seguridad, testing unitario y de integración.

3.3 FODA

Se hizo un análisis de las características del equipo frente a la problemática planteada con el objetivo de saber en qué aspectos internos el grupo era fuerte y débil, y en qué aspectos externos se pueden encontrar oportunidades y amenazas. De esta manera se pudo resumir la situación en la que se encuentra el proyecto, para así poder planificar de forma correcta el proceso de desarrollo y las metodologías a seguir.

3.3.1 Fortalezas

- Solución económica frente a los competidores.
- Se proporcionará una solución modularizada y escalable, cuyo código fuente estará disponible para el cliente, con el fin de poder adaptarlo a sus necesidades en el futuro de ser necesario.
- Brindar información en tiempo real sobre la cantidad de personas en la planta.
- Monitorear la documentación subida a la plataforma mediante inteligencia artificial.
- Los contratistas podrán cargar documentación incluso desde la puerta de la empresa.
- Acceso móvil y web e inmediatez de la información.
- Se utilizarán las últimas tecnologías que permitan la escalabilidad del proyecto en términos de infraestructura y arquitectura.
- La empresa dispone de un área de sistemas, por lo tanto se podrá realizar una transferencia de conocimientos.

3.3.2 Oportunidades

- Se presenta una solución novedosa y atractiva a un problema recurrente en las empresas locales, lo que posibilita la aplicación de dicha solución en nuevos sectores/empresas.
- La empresa ya posee un sistema de control de horarios de los empleados, lo que permite diferenciar el tipo de control del proyecto y de dicho sistema, siendo el nuevo control uno interno y evitando así problemas legales con el mismo.
- Se podría comercializar/escalar el módulo de inteligencia artificial para empresas con problemas/arquitecturas similares.
- Durante la pandemia la empresa tuvo una importante necesidad de contar con el sistema de forma inmediata.

3.3.3 Debilidades

- La falta de experiencia por parte de los miembros del equipo en algunos aspectos puede llegar a presentar dificultades en el desarrollo del proyecto, especialmente a la hora de identificar apropiadamente las anomalías dentro del conjunto de datos.
- El entrenamiento de la red neuronal, necesitará disponer de un volumen grande de datos, para el cual habrá que esperar que el sistema esté instalado y disponga de un periodo aceptable de datos cargados.
- Al trabajar en forma virtual, se dificultó hacer el relevamiento/consultas, debido a que las personas a ser entrevistadas se encuentran ocupadas y/o no disponibles.
- No se conocían los procesos organizacionales de la empresa.

3.3.4 Amenazas

- El sistema puede recibir una fuerte resistencia por parte de los empleados.
- El sistema presenta una solución a un problema frecuente, por lo que pueden llegar a aparecer nuevas soluciones de otros competidores.
- Un nuevo rebrote de COVID podría conducir a un nuevo confinamiento, lo cual podría inducir a la empresa a problemas económicos, entre los cuales se encuentra la quiebra.

3.4 Análisis de benchmarking

Antes de desarrollar el producto se realizó un análisis de los competidores para tener información acerca de sus fortalezas y debilidades. De esta forma, se obtuvo valiosa información no solo para cumplir con las demandas que no están satisfechas, sino también aplicar sus estrategias e ideas para alcanzar un mejor producto final.

Para ello, se realizó un estudio sobre los productos más destacados en el área, obteniendo sus ventajas y desventajas. Ellos son: Nextar, Wilsoft QSupplier y SoftExpert.

3.4.1 Nextar

Es un sistema de gestión de proveedores integrado con los procesos de compras y ventas que facilita la administración de proveedores de manera sencilla y segura.

Ventajas:

- Fácil de utilizar.
- Filtros personalizables.
- Informes por proveedor.
- Manejo de inventario.
- Soporte 24/7.
- Bajo precio.

Desventajas:

- Tiene funcionalidades que al cliente no le interesan (Control de caja, puntos de venta, etc).
- La experiencia de usuario no es buena. Le falta un diseño más amigable e intuitivo.
- No tiene control de acceso a planta.
- No tiene soporte para dispositivos móviles.

3.4.2 Wilsoft QSupplier

Es un software que facilita el control y la evaluación de los proveedores de manera flexible y rápida, permitiendo a la organización desarrollar proveedores con el fin de asegurar la eficacia y eficiencia del proceso de compras.

Ventajas:

- Permite evaluar periódicamente el desempeño de los proveedores.
- Campos y reportes personalizados.
- Fácil de utilizar.
- Análisis con gráficos.
- Soporte para dispositivos móviles.

Desventajas:

- No tiene seguimiento de documentación.
- Al no contar con seguimiento de documentación, el flujo de acceso a planta no podrá cumplir las necesidades de Nutreco.
- Precio elevado.

3.4.3 SoftExpert

Es un sistema de gestión de proveedores que permite la visualización de los datos de los proveedores de forma rápida y fácil.

Ventajas:

- Categorización de los proveedores.
- Indicadores de desempeño y visualización sencilla de los documentos.
- Permite visualizar productos y servicios ofrecidos por el proveedor.
- Permite asignar requisitos específicos a cada proveedor.

Desventajas:

- Diseño aburrido.
- No cuenta con control de acceso a planta.
- No tiene soporte móvil.
- Precio elevado.

3.4.5 Resultados del benchmarking

Este análisis arrojó resultados muy importantes, los cuales fueron utilizados para agregar nuevos requerimientos y mejorar el producto final.

Lo primero a destacar es que el sistema debe contar con un diseño lo más intuitivo posible. De esta forma, el usuario no tendrá que invertir tiempo en aprendizaje y hará que se sienta más cómodo al utilizarlo. Luego, los indicadores en pantallas principales son importantes para tener un panorama general del estado de la información que maneja cada usuario. Por ejemplo: Indicadores con documentos vencidos o próximos a vencer. También los filtros personalizados son de gran utilidad, de esta forma podrán concentrarse en los datos que les interesan. Por último, un aspecto fundamental hoy en día es que el sistema pueda ser utilizado en dispositivos móviles. Obligar al usuario a acceder desde una computadora es limitar el uso de la plataforma.

3.5 Análisis de riesgos

Se confeccionó una lista de riesgos que pueden surgir durante la realización del proyecto junto con su evaluación y se desarrolló una respuesta al mismo.

Riesgo	Impacto	Respuesta
Licencia o vacaciones de un miembro del equipo	Bajo	Todos los miembros deben saber todo el trabajo necesario, de forma tal que alguna baja temporal no impida continuar con el desarrollo del proyecto.
Falta de claridad en los roles del equipo	Bajo	Plantear los roles desde el principio.
Alta complejidad técnica	Medio	Plantear diferentes soluciones viables para un problema de forma que, si se vuelve muy complejo de implementar, se pueda optar por otra solución.
Desconocimiento de las tecnologías	Medio	En lo posible elegir tecnologías conocidas para evitar asignar tiempo en su aprendizaje.
Cambios de requerimientos	Alto	Elegir una metodología de desarrollo que contemple y acepte cambios en los requerimientos.
Estimación inadecuada del tiempo de ejecución	Alto	Aplicar un porcentaje de sobreestimación (10%) para cada tarea.
Falta de claridad en los requerimientos	Alto	Validar con el cliente cada uno de los requerimientos planteados, de forma tal que no haya dudas al respecto.
El cliente pide nuevos	Alto	Renegociar el alcance del proyecto

requerimientos a medida que se desarrolla el proyecto		con el cliente. De esta forma se puede sustituir requerimientos de menor importancia por nuevos para no afectar el esfuerzo y tiempo del proyecto.

3.6 Análisis de viabilidad

Se realizó un estudio acerca del proyecto para valorar si es viable desarrollarlo o no. Para ello se elaboró un cuadro con las tareas con su tiempo. Se agregó un 10% extra por cualquier inconveniente o imprevisto que pueda surgir.

Nº	**Tarea**	**Tiempo Estimado**
1	Análisis de requerimientos del sistema	20 horas
2	Relevamiento de Infraestructura de la empresa	20 horas
3	Diseño de la Interfaz del sistema	20 horas
4	Diseño de la Arquitectura del Sistema	20 horas
5	Diseño de la base de datos	16 horas
6	Armado del ambiente de desarrollo	16 horas
7	Desarrollo frontend	120 horas
8	Desarrollo backend	200 horas
9	Desarrollo de base de datos	16 horas
10	Desarrollo del monitor de documentos con alertas de auditoría	60 horas
11	Desarrollo de la Red Neuronal de Anomalías	40 horas
12	Pruebas	20 horas
13	Implementación del sistema	16 horas
14	Documentación del proyecto	Durante todo el proyecto
15	10% Overhead	60 horas
	Total	644 horas

Según el resultado obtenido, se estimó que el proyecto requeriría de 644 horas de trabajo. Esto equivale a alrededor de 65 semanas considerando una dedicación de 2 horas por día. Sin embargo, el cliente no aceptó esta duración debido a la necesidad de contar con el sistema lo antes posible. En consecuencia, se planteó la necesidad de desarrollar un producto mínimo viable que estuviera funcional para la semana 32, y esta propuesta fue aceptada por el cliente.

Desde el punto de vista económico no se encontró ningún impedimento ya que no se necesitó ningún servicio o componente imprescindible que requiera inversión.

4. Proceso

4.1 Metodología de trabajo

“Las metodologías de desarrollo de software son un conjunto de técnicas y métodos organizativos que se aplican para diseñar soluciones de software informático. El objetivo de las distintas metodologías es el de intentar organizar los equipos de trabajo para que estos desarrollen las funciones de un programa de la mejor manera posible. Desarrollar un producto sin una metodología clara desembocará en un proceso aún más complejo, que conducirá a problemas, retrasos, errores y, en definitiva, un mal resultado final.” [1]

Desde las primeras reuniones con el cliente se notó que la definición de los requerimientos fue muy amplia y poco precisa. El cambio de los requerimientos sobre la marcha suele ser algo muy común en los proyectos, y se debe estar sujeto a ellos. El equipo decidió que el uso de metodologías ágiles iba a ser la mejor opción. Si bien nunca es bueno que ocurran cambios, con este tipo de metodologías se puede minimizar el impacto que tienen.

Luego de investigar las diferentes opciones, se hizo una elección entre dos: *Kanban* y *Scrum.*

Se optó por elegir *Scrum* como la metodología a utilizar. El hecho de que el proyecto cuente con poca definición y alta incertidumbre por parte del cliente fue uno de los motivos más importantes para la elección. Además, el cliente tenía la necesidad de implementar cuanto antes el sistema, entonces con esta metodología se podría obtener MVP para luego seguir desarrollando de forma iterativa e incremental.

Se distribuyeron los roles de *Product Owner, Scrum Master* y del equipo de desarrollo. También se hizo un *backlog* con las épicas que se necesitarían. Estas debían ser funcionalidades transversales a toda la aplicación. Las épicas estarían compuestas por historias, las cuales reflejaron una funcionalidad específica. Cada historia se dividiría por tareas y estas mismas por subtareas.

Los *sprints* contaron con una duración de 2 semanas (10 días). Cada sprint contaba con diferentes ceremonias que tenían un objetivo en concreto:

1. *Planning*: Tenía lugar el primer día del sprint. Se completaba el sprint anterior y se iniciaba uno nuevo. Las tareas no finalizadas se movían al nuevo sprint y se asignaba cada una de las nuevas tareas a los diferentes integrantes. En esta

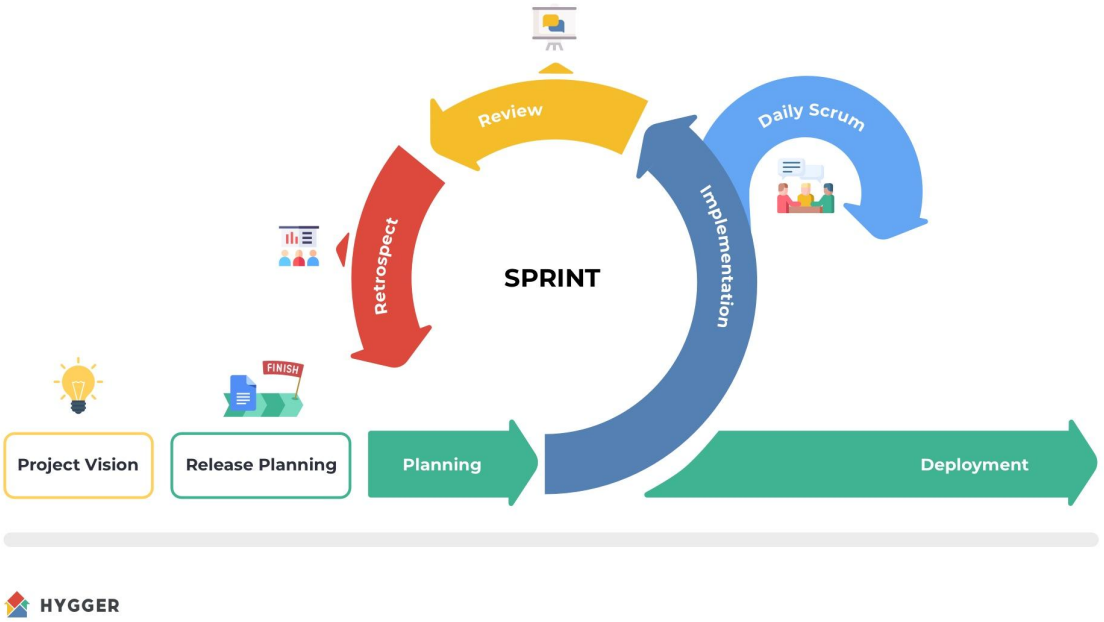
reunión se asignaban y se estiman las tareas a realizar en el sprint. En la asignación se priorizó a los integrantes que ya contaban con experiencia relacionada a la tarea. Respecto a la estimación, se realizó utilizando la escala de fibonacci.

2. *Daily*: Si bien su nombre indica que sería diaria, la realidad es que se limitó ya que 3 por semana era suficiente. En esta reunión se compartía el estado de las tareas que trabajó cada integrante y se informaba en caso de haber algún inconveniente o bloqueante. De esta manera todo el equipo estaba al tanto de lo que estaban realizando sus compañeros.

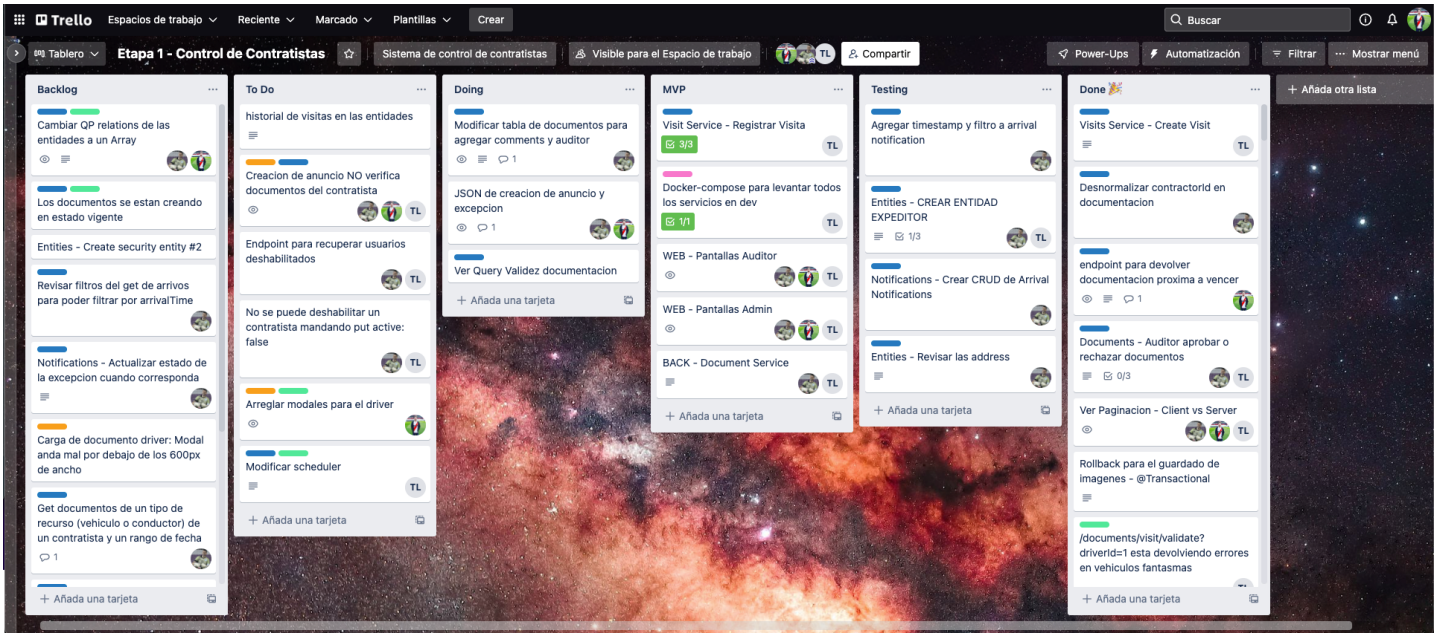
3. *Refinement*: Se llevaba a cabo el 8vo día del sprint y allí se elegían las tareas que se realizarían en el siguiente sprint. Además, se leía la definición de las tareas y se respondían todas las dudas posibles.

4. *Demo*: Entre el 9no y 10mo día se coordinaba una reunión con el cliente para mostrarle los avances. De esta manera el cliente siempre estaba al tanto del avance del proyecto y podía corregir si veía que algo no estaba bien.

5. *Retrospective*: El objetivo de esta reunión era realizar un balance y autocrítica acerca del sprint. Se anotaban las cosas que salieron bien, las que había que mejorar y nuevas ideas para implementar en el proceso. De esta forma se logró aprender para que cada sprint salga mejor.



El tablero utilizado fue un tablero *Kanban* en la plataforma Trello.



Como se ve en la imagen, el tablero consta de diferentes secciones. El ciclo de vida de una tarjeta era el siguiente: Se crea la tarea en la columna *backlog*. Una vez que la tarea es refinada y está lista para ser asignada a un desarrollador, se la mueve a *to do*. Allí se asigna durante la planning. Se mueve a *doing* luego de que el desarrollador la comienza a trabajar y permanece allí hasta que termina y se mueve a testing. Luego de que la tarea se testea, si está aprobada se mueve a *done*, de lo contrario vuelve a *to do* con los comentarios de por que la tarea se rechazó. La columna MVP hace referencia a tareas prioritarias que son necesarias para el MVP. Cabe destacar que el testing era realizado por alguno de los otros integrantes, ya que no es buena idea que la testee el mismo integrante que la desarrolló.

4.2 Gestión de versionado

Como todo proyecto de software, se debe elegir cómo trabajar con el código en forma conjunta y manteniendo un historial de versiones.

Para ello se decidió utilizar GIT. Las grandes ventajas que otorga este software es la capacidad de ramificación. Las ramas principales fueron: *Main* y *Development*. En *Main* se mantuvo el código que estaba ya *deployado* en el servidor del cliente, por lo tanto ante cualquier falla en producción se podía realizar un seguimiento para su rápida solución. En cambio la rama *Development* contenía el código que estaba siendo desarrollado y testeado

ante de enviarse a producción. Para cada nueva tarea se creaba una nueva rama y, una vez finalizada, se mezclaba con el resto del código de la rama *Development* para luego eliminar la rama.

Los repositorios fueron alojados en GitHub. Esta plataforma otorga la posibilidad de configurar varios aspectos de cada repositorio, agregando *issues* o propuestas, *pull requests* para la fusión del código entre ramas y *Actions* que permite la configuración de *CICD* (integración continua y entrega continua).

Este último mencionado (*CICD)* se utilizó para obtener *pipelines* de trabajo una vez que el código era mezclado en la rama *development*. Se configuró para que se realicen los test unitarios, pruebas de *linting* y despliegue automático de los contenedores en los servidores de desarrollo. De esta forma, una vez que el *pull request* era aprobado y aceptado, se contaba con todo el proceso automatizado. Tenía una duración aproximada de entre cinco y diez minutos y, una vez transcurrido este tiempo, el servidor de desarrollo ya contaba con la nueva versión disponible. Se generaba un *artefacto* que sería utilizado luego para su despliegue manual a Producción, el cual era almacenado en la plataforma *Docker Hub* en forma de contenedor.

4.3 Comunicación

En principio, la pandemia limitó la cantidad de reuniones presenciales que se podían realizar, tanto con el cliente como entre los integrantes. Por lo tanto, se realizaron muchas reuniones virtuales a través de Google Meet y Slack. La primera se utilizó principalmente para reuniones directamente con el cliente, en cambio la segunda fue de uso interno del equipo.

Si bien hay muchas herramientas para gestionar el trabajo, se optó por Slack ya que permite integrarse fácilmente con herramientas utilizadas para el desarrollo. Se pudieron recibir notificaciones cuando se creaban *pull request*, se mezclaba nuevo código o se desplegaron nuevas versiones. Además, permite conversaciones tanto orales como escritas.

En cuanto a la comunicación presencial, se lograron concretar visitas a la planta de Nutreco, lo cual hizo que se pueda entender mejor la problemática y plantear soluciones viables. A medida que las restricciones se fueron disminuyendo, se empezó a optar por realizar reuniones presenciales también entre los integrantes.

5. Producto

En el ambiente de la informática, un producto es un software diseñado, desarrollado y mantenido para resolver una necesidad. En esta sección se detallan todas las acciones y decisiones llevadas a cabo para poder obtener el mejor producto en el menor tiempo posible.

5.1 Análisis

Una vez terminada la fase de planificación se dio lugar a la fase de análisis. Es importante entender la problemática para luego elegir la solución más adecuada. Se sabe que el costo de construir un sistema a la primera es mucho menor que construir un sistema que habrá que modificar más adelante. Esta etapa es fundamental, ya que un mal análisis puede generar una solución equivocada o insuficiente. Además, no es una etapa sencilla. En este punto suele suceder que ni el cliente sabe con exactitud qué es lo que quiere y/o necesita.

5.1.1 Elicitación

La ingeniería de requerimientos se trata de recopilar, analizar y verificar las necesidades del cliente para un sistema de software. El objetivo es obtener una especificación de requerimientos de software correcta y completa. Es muy importante entender lo que se va a desarrollar en las próximas etapas. Como dicta el estándar IEEE 830-1998, una buena especificación debe ser completa, consistente, inequívoca, correcta, trazable, priorizable, modificable, verificable y clara.

Se llevaron a cabo muchas reuniones tanto internas como con el cliente. Se trató de obtener la mayor cantidad de información posible en cuanto a la problemática. El equipo utilizó diferentes técnicas para ayudar al cliente a averiguar qué es lo que realmente necesita. Entre ellas se encuentran métodos interactivos como entrevistas y métodos discretos como observación del proceso de trabajo con el que contaba la empresa en ese momento.

Entre los requerimientos, algunos fueron solicitados desde un principio por Nutreco y otros fueron identificados durante esta etapa a través de las diferentes técnicas. Estos son:

Requerimientos Funcionales

Carga de documentación de proveedores

- Interfaz intuitiva
- Permitir sacar foto o escanear
- Campos para cargar datos
- Diferentes modalidades: Por empleado, vehículo, documento, en lote y en formato .zip

Visualización de reportes online

- Acceso web a información actualizada con disponibilidad 24/7 los 365 días del año

Notificaciones

- Emisión de alertas automáticas vía e-mail para seguimiento online, de documentación rechazada o próxima a vencer.

Administrar distintos tipos de usuarios

- Contratista
    - Administrar propios empleados
    - Administrar vehículos propios
    - Realizar consultas sobre el estatus de la información referida a vencimientos y presentaciones de su empresa.
    - Alta, baja y modificación de sus propios recursos.
- Empresa/Admin
    - Alta, baja y modificación de contratistas.
- Encargado
    - Autorizar excepciones ignorando el estado de la documentación.
- Encargado de visitas externas (Este rol fue solicitado en etapas posteriores)
    - Agregar y eliminar visitas de empleados tercerizados.
- Auditor
    - Validar o rechazar la información ingresada, incluyendo comentarios.
    - Parametrizar documentación de contratista
- Seguridad
    - Autorizar ingreso a planta a empleados y contratistas.
    - Conocer si la persona está autorizada o no a ingresar.

    - Registro de ingreso para saber quienes están dentro de la planta.
- Expedición (Este rol fue solicitado en etapas posteriores)
    - Visualizar cola de arribos.
    - Permitir o denegar el ingreso de los conductores que se encuentran en espera, dejando constancia de la razón por la cual se decidió dicha acción (similar a la auditoría de la documentación).

Registrar todas las acciones del contratista en un log

Impresion de codigos QR para el acceso a las instalaciones

- Acceso a cámara de seguridad para escanear
- Generar QR único por proveedor

Requerimientos No Funcionales

- Compatibilidad con PC y smartphones (Android - IOS)
- Detectar anomalías dentro de los horarios de ingreso y egreso de los empleados.
- Facilidad de uso para los usuarios
- Escalabilidad.

Identificar a los diferentes *stakeholders* también fue parte de esta etapa. Se debía conocer los diferentes actores con los que cuenta el proceso. Gracias a esto se pudo detallar los diferentes usuarios con los que contaría el sistema:

- Encargado: Responsable de la autorización de excepciones y generación de reportes.
- Administrador: Tiene acceso total al sistema. Es el único usuario encargado del alta, baja y modificación de los contratistas.
- Auditor: Responsable de la aprobación de los documentos.
- Conductor: Quien maneja el vehículo hacia las instalaciones de la empresa.
- Vehículo: Existen diferentes tipos de vehículos: Con acoplado, semi acoplado, saider, térmico.
- Seguridad: Encargado del ingreso y egreso de la planta.
- Expedición: Responsable de autorizar el ingreso a planta de los vehículos anunciados.
- Encargado de visitas externas: Quien se encargaría de cargar las visitas externas

que puede recibir la planta.

5.1.2 Proceso de negocios

La empresa contaba con un modelo de negocios bien definido. Gracias a reuniones y observaciones en planta se logró plasmar el modelo en diferentes diagramas que facilitan su comprensión. Se desarrollaron diagramas de componentes, de contexto, de transición de estados, de entidad-relación, de casos de uso y de secuencia.

5.1.3 Análisis Técnico

Se realizó un análisis técnico de la infraestructura, conocimientos y experiencia del cliente para ayudar a la elección de tecnologías, protocolos, prácticas, etc. Las decisiones se basaron principalmente en la necesidad del cliente y la velocidad de implementación, ya que el cliente tenía una clara urgencia de contar con el sistema lo antes posible. La experiencia previa del equipo fue importante a la hora de elegir entre opciones igualmente válidas.

Infraestructura

Tras una serie de reuniones, sumado a la visita a la planta, se logró hacer un resumen sobre la infraestructura con la que contaba el cliente.

- **Servidor:** La empresa ya tenía con un servidor propio que estaba disponible para que se pueda correr el sistema sin problemas. El sistema operativo era Linux.
- **Red:** La conexión a internet estaba disponible en todos los lugares que se necesitaban, ya sea de forma inalámbrica o por cable.
- **Sistemas Operativos:** Principalmente Windows y Linux eran los sistemas operativos que estaban instalados en las computadoras del cliente. Para los dispositivos celulares solamente se encontró instalado Android.
- **Base de datos:** El área de sistemas de Nutreco ya disponía de una base de datos en *SQL Server* con licencia paga. Este dato no es menor, ya que permitió saber en qué tipo de bases de datos tenían experiencia.
- **Nube:** La empresa no utilizaba la nube.

Lenguajes

A la hora de indagar respecto a la experiencia en lenguajes de programación con la que contaba el personal del área de sistemas de Nutreco se encontró con que los lenguajes predominantes eran PHP, Java y Javascript en el *backend* y Javascript en el *frontend.*

Aun así, el cliente dió total libertad de los lenguajes a utilizar, a pesar de que el mantenimiento del sistema quedaría a cargo de ellos. El motivo fue que no se quiso limitar la elección de lenguajes para poder utilizar los que mejor se adapten a las necesidades.

5.2 Diseño

"Se lo define como el proceso de aplicar ciertas técnicas y principios con el propósito de definir un dispositivo, un proceso o un sistema, con suficientes detalles como para permitir su interpretación y realización física." [2]

En esta etapa se logró definir las diferentes soluciones posibles al problema y elegir la más adecuada. Además, se tuvo en cuenta el proceso de negocios analizado en la etapa anterior con la finalidad de evaluar mejoras y nuevas funcionalidades que sean de utilidad. Es una buena etapa para redefinir flujos con el objetivo de solucionar problemas o incrementar la *performance* del proceso.

5.2.1 Arquitectura

"La arquitectura de software se refiere a las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos." [3]

5.2.2 Backend

Luego de analizar las diferentes opciones, se optó por una arquitectura basada en microservicios ya que se adapta mejor a las necesidades del cliente. Permitió agilizar el proceso de desarrollo al tratarse de servicios autónomos se pueden desarrollar y desplegar de forma independiente. Además, la versatilidad al momento de elegir diferentes tecnologías y el lenguajes de programación que mejor se adapten a cada servicio facilitaba

la integración del servidor web junto con las inteligencias artificiales. Sumado a esto, se contaba con la posibilidad de migrar el sistema a un entorno *cloud* una vez finalizado el desarrollo, por lo que esta arquitectura resulta más amigable en términos de mantención y portabilidad.

Dentro de los diferentes lenguajes de programación para el uso en la creación de un servidor web, como Java, PHP, Python y Javascript, se decidió utilizar este último por su versatilidad, popularidad entre la comunidad por ser de código abierto y por ser un poderoso lenguaje del lado del servidor al ser combinado con la tecnología de Nodejs.

Microservicios

Para la implementación de los microservicios se eligió utilizar contenedores por sobre máquinas virtuales. Los contenedores son una forma de virtualización, los cuales no contienen una imagen del sistema operativo lo que los hace más rápidos y livianos, es decir, dentro de un contenedor solo se encuentran el código binario, las bibliotecas y archivos de configuración necesarios para ejecutar una aplicación. Esto permite que se puedan correr fácilmente un grupo de contenedores en una misma máquina host, haciendo que su uso sea más eficiente. Gracias a los contenedores, se pudo crear un ambiente aislado y escalable para cada microservicio.

Tras una evaluar diferentes opciones, se llegó a la conclusión de que la mejor opción para el desarrollo, implementación y ejecución de las aplicaciones en los contenedores era Docker. El cual, en pocas palabras, permite crear una imagen (docker image) con una serie de instrucciones para poder ejecutar un contenedor (docker container) con la configuración necesaria para que cada microservicio funcione correctamente en su propio ambiente con límites bien definidos.

Framework

Los frameworks para Nodejs más utilizados y que a su vez tengan como principal objetivo la creación de *REST APIs* son: *Express*.js y *NestJS*

Express.js

Express es un *framework* que provee un amplio rango de funcionalidades para el desarrollo de aplicaciones *web* y móviles. Soporta el uso de patrones de diseño MVC: Modelo, Vista y Controlador.

Ventajas:

- Mayor popularidad y comunidad.
- Minimalista y liviano.
- Fácil aprendizaje.
- Flexible.

Desventajas:

- No es fácilmente escalable.
- No soporta Typescript, haciendo que sea menos eficiente y poco compatible con múltiples navegadores.

NestJS

NestJS es utilizado para la construcción de aplicaciones del lado del servidor. Está basado en Typescript e inspirado en Angular.

Ventajas:

- Ofrece simplicidad para el desarrollo de microservicios.
- Es escalable y mantenible.
- Extensa documentación.
- Posee una estructura modular, que facilita la división del proyecto en bloques separados. Se adhiere al paradigma de “convención sobre configuración”, el cual intenta disminuir la cantidad de decisiones que un desarrollador debe tomar.
- Contiene Inyección de dependencias integrado. Este es un patrón de diseño que es utilizado para hacer las aplicaciones más eficientes y modulares. Mantiene el código limpio, fácil de leer y utilizar.

Desventajas:

- No posee una amplia comunidad.
- Escalonado: En NestJS, la extensa estructuración de los archivos puede generar que la navegación por los diferentes directorios pueda ser tediosa.

- La curva de aprendizaje exponencial: Al principio cuesta mucho entender cómo funciona este framework, lo que provoca que los desarrolladores requieran de mayor tiempo para cumplir con las tareas y requisitos.

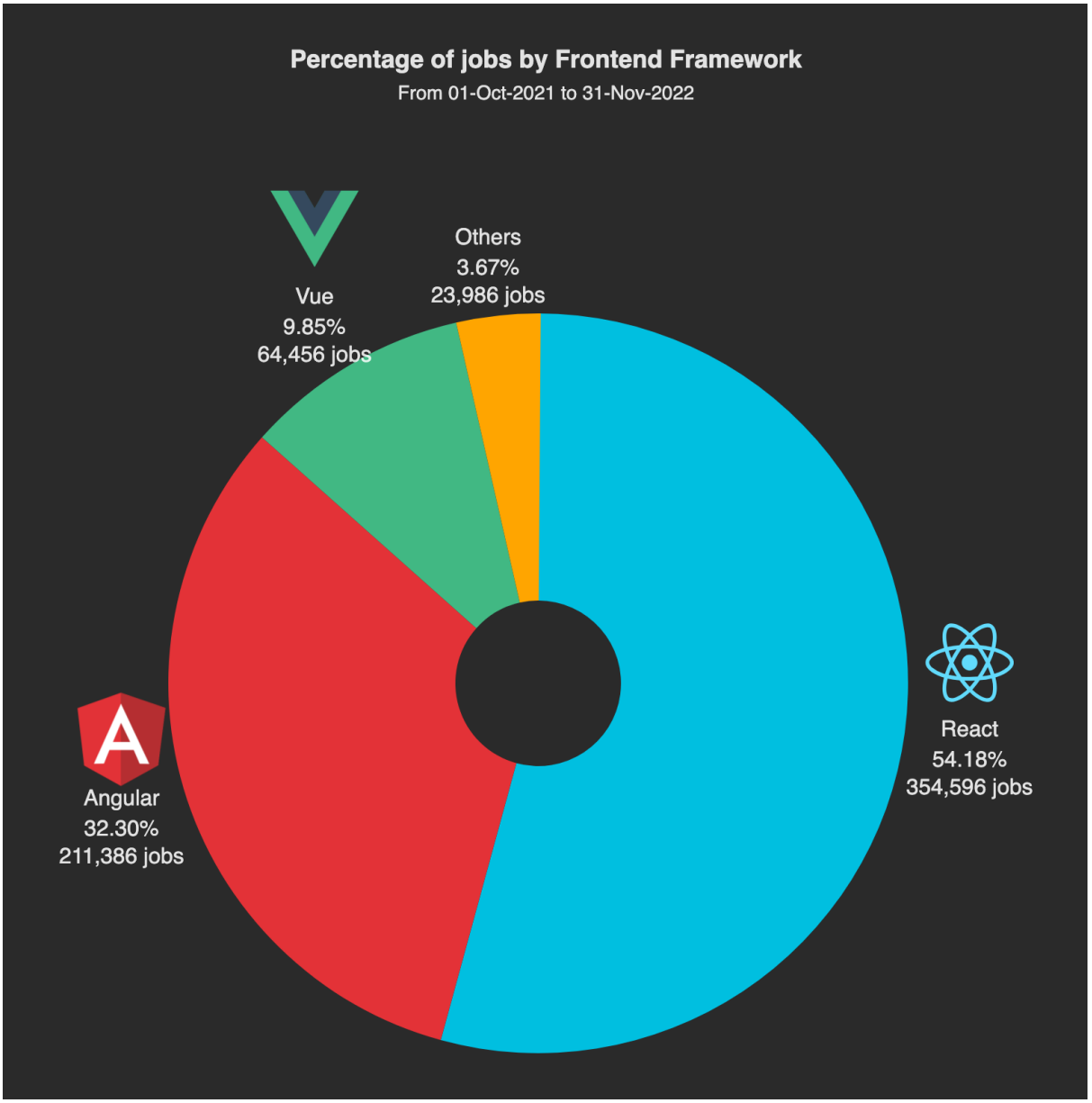
Debido a su escalabilidad, estructuración modular y compatibilidad para el desarrollo de microservicios se optó por el uso de *NestJS* como framework para *Nodejs*. Por lo que se debió enfrentar el desafío de aprender a utilizar esta tecnología para lograr implementarla correctamente. A su vez, su compatibilidad con Typescript permitía que su desarrollo fuera más eficiente y mantenible.

5.2.3 Frontend

Lo más conveniente fue utilizar *HTML, CSS* y *Javascript.* Esto se debe al previo conocimiento de los empleados de Nutreco y a la gran comunidad y popularidad que poseen. Además, *Javascript* es un lenguaje muy sencillo, veloz, versátil, con gran soporte para los navegadores, ya sea en versión de escritorio o móvil. Otro adicional fue la elección de *Javascript* en *backend,* disminuyendo la cantidad de lenguajes y facilitando el trabajo en modo *Full Stack.*

Frameworks y librerías

Los frameworks y librerías para *Javascript* más utilizados en el momento son: *Angular, React.js* y *Vue.js*



[4]

Angular

Angular es un framework JavaScript muy popular del tipo Modelo-Vista-Modelo de vista, creado en 2009 y perfecto para construir aplicaciones web interactivas.

Ventajas:

- Compilado más rápido.
- Gran comunidad y documentación.
- Vínculo de datos en ambos sentidos.
- Permite al desarrollador trabajar en la misma sección de la aplicación usando los mismos datos gracias al patrón MVVM (Modelo - Vista - Modelovista).

- Inyección de dependencias de las características relacionadas a los componentes de los módulos y la modularidad en general.

Desventajas:

- Pueden existir problemas de migración entre versiones.
- La curva de aprendizaje es lenta.
- Sintaxis compleja heredada de la primera versión de Angular.

React.js

ReactJS es una librería JavaScript de código abierto, creada por Facebook en 2013, que es perfecta para construir aplicaciones web grandes donde los datos se modifican con regularidad.

Ventajas:

- Fácil de aprender.
- Gran nivel de flexibilidad y máxima “responsabilidad”.
- DOM Virtual (Modelo de Objetos del Documento) que permite convertir documentos HTML, XHTML o XML en un árbol más fácil de manejar para los navegadores. De esta forma se evitan re-renderizados innecesarios de la página al realizar comparaciones entre el DOM Virtual y el DOM, de forma tal de solamente actualizar los nodos que cambiaron.
- Vínculo de datos “hacia abajo”. Los datos de los elementos “hijos” no pueden afectar a los “padres”.
- Librería 100% de código abierto.
- La migraciones entre versiones son muy sencillas.
- Comunidad muy grande.

Desventajas:

- Ausencia de documentación oficial. Aunque al tener una comunidad tan grande, la falta de información no es un problema.
- No existe un estándar de desarrollo. Se puede resolver el mismo problema de múltiples formas diferentes.

Vue.js

Vue.js es un framework JavaScript, lanzado en 2013, que encaja perfectamente en la creación de aplicaciones con interfaces de usuario altamente adaptables y del tipo Single-Page (Página Única).

Ventajas:

- HTML empoderado. Esto significa que Vue.js tiene muchas similitudes con Angular y esto puede ayudar a optimizar el manejo de bloques HTML usando diferentes componentes.
- Documentación detallada.
- Adaptabilidad.
- Excelente integración. Vue.js puede usarse para construir tanto aplicaciones de página única o complejas interfaces web de aplicaciones.
- Gran escalabilidad.
- Pequeño tamaño.

Desventajas:

- Riesgo de excesiva flexibilidad.
- Falta de documentación completa en inglés.

Finalmente, luego de evaluar las diferentes alternativas, se decidió que *React.js* es la mejor opción. Es el que menos tiempo le llevaría aprender a los empleados del área de sistemas de Nutreco y, en caso de incorporar personal para su mantenimiento, es el que domina el mercado laboral por muchísima diferencia. Además, tiene su contraparte móvil llamada *React-Native,* la cual sería muy útil en caso de necesitar construir una aplicación para Android e IOS*.* Si bien no son exactamente iguales, son bastante similares. Por lo tanto, aprendiendo uno sería muy fácil aprender el otro.

5.2.4 Mobile

Uno de los requerimientos solicitados por el cliente fue que los usuarios, en especial el conductor, pueda acceder mediante dispositivos celulares al sistema. Esto se debía a que es muy común que les falte alguna documentación al llegar a la planta. La idea es que puedan cargar la misma desde el celular subiendo la foto del documento.

Para ello se presentó al cliente dos alternativas: Desarrollar una aplicación nativa o una progressive web app (PWA).

Ventajas de PWA

- Instalación: No necesita ser instalada en el dispositivo para que los usuarios la utilicen. Además, puede simular una instalación permitiendo agregar un acceso directo a la home screen del dispositivo, casi sin ocupar espacio.
- Velocidad de Carga: El uso de Service Workers, al cachear recursos, les otorga a las PWA una velocidad de carga superior a las aplicaciones nativas.
- Precio y tiempos: Se ahorran los costos de las licencias y se ahorran los tiempos de creación de cuentas para subir aplicaciones tanto a AppStore como Play Store. Esto es una ventaja ya que la burocracia que ello implica puede resultar un proceso tedioso y de varios meses.
- Desarrollo: La aplicación puede desarrollarse en el mismo lenguaje que las demás webs previstas, es decir, no requiere de un lenguaje específico para dispositivos móviles. Esto no sólo ahorra tiempo en el desarrollo, sino también en su mantenimiento y deploy, ya que el cliente no cuenta con los dispositivos de Apple necesarios para subir los archivos .ipa a la AppStore.

Ventajas aplicaciones nativas

- Compatibilidad: No todos los navegadores soportan el uso de PWA. Sólo las versiones más recientes (mayores a 2018) de Chrome, Firefox y Opera lo soportan.
- Consumo de Batería: Las aplicaciones nativas consumen menos batería que las PWA ya que estas últimas corren sobre un navegador web y no hay un control directo sobre el consumo de la batería. Mientras que el consumo y eficiencia de las app nativa están bajo el control del sistema operativo donde corre.
- Distribución y aceptación de los usuarios: En el caso de las aplicaciones nativas, sólo basta con subirlo a una tienda para que todos los usuarios puedan descargarla y tener el acceso directo en su celular. En cambio, crear un acceso directo para una aplicación progresiva, si bien es un proceso sencillo no es algo familiar para todos los usuarios, y puede depender del Sistema Operativo del dispositivo y de su versión.
- Funcionalidad: Las PWA tienen un uso más limitado de funcionalidades nativas, como por ejemplo, uso de bluetooth, NFC, acceder a la lista de contactos, etc. Esto

puede llegar a ser un punto clave dependiendo de las características de la aplicación.

Costos de subir una aplicación nativa a las tiendas:

Android: Pago único de licencia de 25 USD.
iPhone: Licencia anual de 99 USD.

El cliente se inclinó por el desarrollo de la PWA principalmente por los costos y tiempos que conlleva subir la aplicación a las tiendas digitales.

Usuario seguridad

Este rol tiene una particularidad diferente al resto de los demás: Se debe dar el ingreso y egreso de los recursos. Para ello, se pensó en un escaneo de códigos generados por el sistema.

Las posibilidades planteadas fueron instalar hardware para el escaneo de los códigos o utilizar la cámara de un celular que utilizaría el personal de seguridad. Por temas de costos y facilidad para el desarrollo, se optó por utilizar la cámara del celular.

Se decidió que era más conveniente desarrollar una pequeña aplicación nativa en la cual se pueda:

- Ver qué recursos se encuentran dentro de la planta.
- Escanear los códigos QR de los conductores para el ingreso.
- Dar egreso a los recursos una vez que se van de la planta.
- Ver y editar datos del usuario de seguridad que está logueado.

El motivo de esta decisión fue que las aplicaciones nativas tienen mejor control sobre el hardware del dispositivo (la cámara en este caso). Además, no sería necesario subirla a las tiendas digitales debido a que solamente la utilizará el personal de seguridad, entonces Nutreco se encargaría de su distribución.

Como se indicó anteriormente, la aplicación se realizó en *React Native* y los archivos .apk fueron enviados al cliente mediante Google Drive para que pueda instalarlos en los dispositivos. La aplicación fue utilizada solamente en dispositivos Android, pero se utilizó un

framework que también soporte IOS para evitar tener que desarrollar otra aplicación si a futuro se decide utilizar dispositivos de Apple.

5.2.4 Routing Web

El sitio web es el mismo para todos los usuarios. Cuenta con un login único que luego redirige al panel correspondiente según el rol. Se desarrollaron diferentes rutas para dividir la información que puede ver y editar cada usuario.

Rol	**Ruta**
Todos los roles	/login
Contratista	/contratista
Auditor	/auditor
Conductor	/conductor
Expedición	/expedición
Encargado	/encargado
Encargado Visitas Externas	/visitas-externas
Seguridad	/seguridad
Admin	/admin

Cada ruta contiene las diferentes subrutas según las pantallas que ese rol puede acceder.

Subruta	**Roles que pueden acceder**	**Descripción**
/contratistas	Admin	ABM sobre una lista contratista.
/conductores	Admin y contratista	ABM sobre una lista de conductores. El contratista solo puede ver los que le pertenecen.
/vehiculos	Admin y contratista	ABM sobre una lista de

		vehículos. El contratista solo puede ver los que le pertenecen.
/auditar	Admin y auditor	Lista y evaluación de nuevos documentos cargados.
/documento	Admin, contratista, auditor, conductor, manager y seguridad.	Lista de documentos de un recurso o contratista.
/users	Admin	Lista de usuarios especiales del sistema (auditor, seguridad, encargado y expedición)
/excepciones	Admin y encargado	Lista y evaluación de excepciones.
/reportes	Admin y encargado	Pantalla para la generación de reportes de permanencia.
/anuncios	Admin y expedición	Pantalla de los anuncios activos al momento.
/historial	Admin y expedición	Listado histórico de anuncios generados con filtros por fecha.
/visitas-externas	Admin y Encargado Visitas Externas	Pantalla para la creación y eliminación de visitas externas

5.2.5 Base de datos

Para este proyecto se ha decidido utilizar bases de datos relacionales. Esto se debe a que evitan la duplicidad de registros, garantizan la integridad referencial, es decir, al eliminar un registro se eliminan todos los registros relacionado dependientes y, favorece la normalización y desmalizacion por ser más comprensible y aplicable.

Dentro de las bases de datos relacionales se optó por utilizar *MySQL* ya que es de código abierto, por lo tanto no implicaría un gasto extra de licencia a Nutreco. Además, es fácil de configurar y requiere poco ajuste para lograr excelentes niveles de rendimiento. Tiene muy buena compatibilidad con diferentes sistemas operativos y una gran comunidad.

Durante el diseño se elaboró el diagrama de entidad-relación de la base de datos para luego continuar con los diagramas específicos de las base de datos de cada microservicio.

5.2.6 Servidor

Si bien el cliente desde un principio tuvo la idea de alojar el sistema en sus servidores, se le presentó una propuesta para utilizar un servidor *cloud* o de nube. Un servidor *cloud* es un recurso de servidor centralizado y agrupado que se aloja y distribuye a través de una red (generalmente Internet) y al que pueden acceder múltiples usuarios cuando lo necesiten.

Las ventajas que proporciona un servidor *cloud* son:

- **Asequibilidad**: A la empresa le sale mucho más barato alquilar un servidor *cloud* que comprar y mantener un servidor propio de las mismas características.
- **Comodidad:** Los recursos en la nube se pueden aprovisionar en pocos minutos y se pueden gestionar fácilmente a través de un panel de control o una API.
- **Escalabilidad:** Esta es una de las grandes ventajas que proporcionan. Los servidores de nube se pueden adaptar fácilmente cuando es necesario cambiar recursos informáticos y de almacenamiento de datos.
- **Fiabilidad:** Otra gran ventaja es que, además de proporcionar el mismo rendimiento que los servidores dedicados, se ejecutan múltiples servidores de un entorno compartido. Esto significa que podrá seguir funcionando aunque falle un componente individual.

Se analizaron las siguientes alternativas de servidores Cloud, comparando los diferentes servicios que ofrece cada plataforma y sus precios:

	DigitalOcean	Google Cloud Platform	Microsoft Azure
Compañía Matriz	Independiente	Google	Microsoft
Modelo de precios	Precios con todo incluido	Precio por servicio	Precio por servicio
SLA (Garantía del	99.9%	99.9%	99.9%

nivel de servicio)			
Regiones y zonas de disponibilidad	EE.UU, Europa e India - 8 ubicaciones	EE.UU, Europa, América del Sur, India, Australia y Asia Pacifico - 25 regiones	América, Europa, Oriente Medio, África, Asia Pacifico - 28 regiones
Facilidad de uso	Muy fácil	Fácil	Intermedio
Precio	Menos costoso	Más costoso	Más costosos
Facturación	Por hora	Por segundo	Por segundo
Créditos	$100 para nuevos usuarios	$300 para nuevos usuarios	$200 para nuevos usuarios dentro de los primeros 30 días
Características	Menos completo	Más completo	El mas completo
Seguridad	Muy buena	Muy buena	Muy buena

Google Cloud Platform

Es una excelente opción para pequeños sistemas y empresas. Tiene millones de usuarios y muchas características sobresalientes que hacen que esta plataforma sea segura, escalable y de buen rendimiento. Además, poseen un excelente equipo de soporte para ayudar con cualquier problema que se presente mientras se crea la infraestructura del sistema.

En cuanto a precios, con la calculadora de precios provista por Google se obtuvo un estimativo desde 25U$D a 50U$D por mes.

Microsoft Azure

La principal ventaja de Microsoft Azure es su velocidad. Esta plataforma hace de los despliegues, recuperación de datos y la escalabilidad un proceso realmente rápido. La cantidad de servicios disponibles para los usuarios es extraordinaria, ofreciendo además una excelente integración con otros productos de Microsoft.

Los precios de Azure son competitivos. Una máquina virtual promedio de Linux (0.75GB, 1 núcleo, 20GB de disco no sólido) cuesta alrededor de 15U$D por mes.

DigitalOcean

DigitalOcean es una plataforma prestadora de servicios en la nube que compite con grandes empresas como las mencionadas anteriormente. Sus principales ventajas son: sus bajos costos, altos rendimientos en máquinas virtuales y simplicidad.

Los precios de DigitalOcean son muy asequibles, no generan gastos extras por cantidad de tráfico o direcciones IP estáticas. Su máquina virtual o *Droplet* más popular (1GB de memoria, 1 núcleo de procesamiento, 30GB de disco SSD) cuesta alrededor de 10 U$D por mes.

Decisión

Después de presentar varias opciones, al cliente le pareció interesante la posibilidad de utilizar un servidor en la nube. Sin embargo, los costos de las plataformas eran altos para hacer el cambio a corto plazo. Por lo tanto, el cliente optó por seguir usando sus propios servidores alojados en la plataforma de Microsoft Azure. A pesar de esta decisión, el equipo no solo se encargó del desarrollo del sistema para que fuera portable y compatible con todas las plataformas en la nube evaluadas, sino que también se encargó de la configuración, mantenimiento y monitoreo del sistema en los servidores del cliente.

Aunque la decisión del cliente no tuvo un impacto directo en el proyecto, no se pudo aprovechar los servicios proporcionados por las plataformas en la nube, como alta disponibilidad, balanceo de carga y escalabilidad. De todas formas, es posible realizar una migracion facilmente en etapas posteriores.

Servidor de desarrollo

Fue necesario la utilización de un servidor en la nube para el momento de desarrollo. Tener un ambiente en el cual puedan realizarse pruebas mientras se construye el sistema es de mucha utilidad. Su objetivo es ayudar a los desarrolladores a que suban las nuevas funcionalidades sin necesidad de pasar por testings de QA o QC. De esta manera, las nuevas funcionalidades están al alcance del resto del equipo.

El servidor constó de una máquina virtual con sistema operativo *Ubuntu* (Distribución basada en Debian GNU/Linux) donde se contaba con las siguientes tecnologías:

- MySQL: Este sistema de gestión de bases de datos relacionales fue el elegido para llevar a cabo la creación de la base de datos, así como las tablas, store procedures y triggers necesarios.
- Docker: Se utilizó esta herramienta para el despliegue de los microservicios, ya que permite realizarlo dentro de contenedores de software. Además, facilita la integración continua de código y el despliegue continuo de la aplicación.
- NGINX: En este caso, se utilizó este servidor web/proxy inverso para servir las páginas web estáticas del frontend como también para responder a las peticiones HTTP del cliente, redireccionandolas al microservicio correspondiente.

Se barajaron las diferentes opciones mencionadas anteriormente, pero finalmente se optó por alojar el servidor en *DigitalOcean* debido a su bajo costo y su sencilla implementación.

5.2.7 Comunicación entre servicios

APIs

Al contar con una arquitectura de cliente-servidor con microservicios fue fundamental contar con APIs bien definidas. El hecho de definir con anticipación las que se iban a necesitar ayudó mucho al desarrollo. La definición del contrato entre *frontend* y *backend* permite saber qué datos se van a enviar y recibir, de forma tal que ambos desarrollos pueden concurrir en paralelo y así ahorrar tiempos.

Lo primero que se debió decidir fue si se utilizaría REST o GraphQL como protocolo de comunicación de la API. Para ello se hizo una investigación acerca de las ventajas y desventajas de ambos:

REST

Ventajas:

- Tiene mejor rendimiento y escalabilidad.
- Cuenta con gran madurez en el mundo tecnológico. La gran mayoría de los desarrolladores ya están familiarizados con el.
- Es fácil de aprender.

Desventajas:

- Requiere que solicite todos los datos debajo del recurso antes de recorrerlo para obtener los datos que necesita, lo que a menudo resulta en grandes cargas útiles.
- Se deben tener diferentes *endpoints* para las peticiones.

GraphQL

Ventajas:

- Permite al cliente pedir exactamente los datos que necesita. De esta forma se evita tener datos innecesarios o inútiles almacenados.
- Facilita la comunicación del equipo al exponer todos los métodos de la API en forma sencilla.
- Es un lenguaje tipado.
- Mejor performance.

Desventajas:

- No posee una amplia comunidad ni documentación, ya que es una tecnología nueva.
- Es difícil de implementar, sobre todo en el lado del servidor. Además su curva de aprendizaje es más pronunciada.
- Tiene graves problemas de almacenamiento de caché (específicamente el almacenamiento en caché HTTP).
- Solicitar demasiados campos anidados a la vez puede generar consultas circulares y bloquear el servidor.
- No admite carga de archivos de fábrica y requiere una solución alternativa para esta funcionalidad.

El equipo comenzó a desarrollar utilizando *GraphQL* debido a la mejor performance que provee. Pero, al cabo de una semana de implementación, la poca documentación y comunidad que tiene hizo que sea cada vez más difícil avanzar. Además, teniendo en cuenta el previo conocimiento de los empleados del área de sistemas de Nutreco, sumado a la experiencia del equipo y la velocidad de implementación, se decidió cambiar a APIs con REST. Esto acortó los tiempos de desarrollo y eliminó los tiempos de aprendizaje necesarios.

5.2.8 Inteligencia Artificial

Se sabía que dos de los requerimientos iban a requerir soluciones utilizando inteligencia artificial. Una de ellas, la detección de anomalías en el horario de ingreso y egreso de los empleados de la empresa fue sustituida por pedido del cliente ya que surgieron nuevos requerimientos con mayor prioridad, así que se pospuso para el futuro. La segunda era una solución para el monitoreo y validación de los documentos previo a la auditoría del personal de la empresa.

En principio se pensó que el problema sería resuelto con una red neuronal, específicamente a través de *deep learning*. Se trata de un tipo de proceso de machine learning llamado aprendizaje profundo que utiliza los nodos o las neuronas interconectados en una estructura de capas que se parece al cerebro humano. Luego de investigar al respecto, no se optó por entrenar una red neuronal, sino que se decidió utilizar una tecnología que permite la digitalización de texto impreso y su conversión en texto digital llamada *OCR* (Reconocimiento Óptico de Caracteres). Esta tecnología utiliza algoritmos de inteligencia artificial y procesamiento de imágenes para analizar imágenes de documentos escaneados y reconocer los caracteres y símbolos en ellos. A su vez, se utilizó el algoritmo de Levenshtein para calcular la cantidad mínima de operaciones necesarias para transformar una cadena de caracteres en otra, y, de esta forma, aumentar la probabilidad de encontrar el texto esperado en la imágenes según el tipo de documento.

Esta tecnología se aprovechó para facilitar la tarea del Auditor. Cada vez que se adjunta un documento que está incompleto o con baja calidad, el *OCR* genera una advertencia al usuario para que adjunte otra imagen. Por pedido del cliente, esto no bloquea el proceso de cargar documentación, de forma que es responsabilidad del usuario si desea continuar la carga de un documento que no está en las mejores condiciones.

6. Memorias

Durante la realización del proyecto han ocurrido muchos sucesos que afectaron en mayor o menor medida al equipo, al cliente y al proyecto mismo. Esto provocó cambios en los tiempos de entrega, requerimientos y desarrollos.

6.1 Trabajo en equipo

Para el desarrollo del proyecto fue fundamental el trabajo en equipo. Los estudiantes, las diferentes áreas de Nutreco (Sistemas, Recepción y Operaciones) y el director del proyecto trabajaron en conjunto a través de reuniones, llamadas y emails.

El equipo comenzó el proyecto con solamente dos integrantes, Martín e Ignacio. Se trabajó con el cliente en los primeros requerimientos y posibles soluciones que se podrían brindar. Luego, se decidió incorporar a un tercer integrante: Tomas. El equipo consideró que su ingreso aportaría valor al proyecto, además de ser un compañero y amigo durante varios años, por lo que también sumaría a la química del equipo.

El área de sistemas de Nutreco fue quien condicionó los requerimientos técnicos ya que contaban con mayores conocimientos en informática. Pero luego hubo que trabajar con el área de usuarios del sistema para los requerimientos funcionales. Eran quienes iban a hacer uso del sistema y no hay nada mejor que tratar directamente ellos.

Para el testing también se trabajó en conjunto. Para cada funcionalidad, el equipo realizó un primer testing al finalizar el desarrollo. Luego, el área de sistemas de la empresa hizo un testing integral que servía para encontrar *bugs* no detectados hasta el momento y para validar que la experiencia de usuario fuera la esperada.

El director del proyecto guió al equipo en todo momento a través de diferentes medios. Su rol fue importante para evacuar dudas y agregar detalles que el equipo no notó debido a la falta de experiencia.

6.2 Obstáculos

El primer obstáculo fue la pandemia del COVID-19. Durante los primeros meses se dificultaron los encuentros presenciales, algo muy importante durante la etapa de análisis. Si bien, como se mencionó en el capítulo 4, las entrevistas y la observación en vivo de los procesos de la planta para la elicitación de requerimientos se pudieron llevar a cabo, estuvieron muy limitadas. Se hubiera deseado poder realizar mayor cantidad de visitas a la planta. También afectó a las reuniones internas del equipo. Al principio solamente se pudieron realizar reuniones virtuales, pero el proyecto comenzó a avanzar con mayor velocidad cuando se pudieron concretar reuniones presenciales.

Otro obstáculo fue la demora que existió entre la entrega de la primera versión del sistema hasta la utilización del mismo en el ambiente de producción. Por cuestiones internas de la empresa se pospuso varios meses la puesta en marcha. La más destacada de estas cuestiones fue un *hackeo* que sufrió la empresa Nutreco a sus servidores. Esto resultó un retrasó de muchos meses ya que no solo se habían vulnerado los servidores donde se alojaba el sistema de este proyecto, sino todos los otros servidores con los que contaban. Esto fue algo totalmente negativo para el proyecto ya que se necesitaba de su uso para generar los datos que luego servirían como entrenamiento para las redes neuronales.

Tras esto hubo mucha incertidumbre respecto al proyecto, ya que generó resistencia dentro de la empresa a compartir información. Nutreco hizo un recálculo de prioridades que no solo demoró el proyecto, sino que hasta hubo riesgo de que se abandonara de parte del cliente. Por fortuna, se decidió continuar con el mismo.

Sumado a lo anterior, se puede agregar que los integrantes del equipo no pudieron dedicarle al proyecto la totalidad del tiempo necesario ya que cada uno contaba con un empleo. Si bien esto dilató los tiempos de desarrollo, también fue algo positivo debido a que los estudiantes ganaron mucha experiencia y conocimiento que luego impactaron directamente en la calidad del producto.

Por último, los cambios de requerimientos solicitados por la empresa también fueron un problema. La mayoría se produjeron luego de las etapas de análisis y diseño, y algunos hasta en una etapa de desarrollo avanzada. El hecho de tener que volver atrás con el desarrollo para realizar un análisis y diseño de un nuevo requerimiento implica un esfuerzo

mayor que si se hubiese planteado desde el inicio. Estos desvíos serán tratados en la sección 6.4.

6.3 Cumplimiento y evolución de los objetivos

Desde un principio se plantearon diferentes objetivos para el proyecto. En esta sección se detalla cada uno junto con evolución.

Objetivo 1: Informatizar y automatizar los procesos de control de la documentación de los contratistas e intercambio de información con los transportistas, a través de un sistema que permita a los mismos el ingreso de dicha información de forma rápida, fácil y eficiente, lo cual permitirá agilizar el proceso de validación de la misma.

Fue el objetivo central del proyecto y requerimiento principal del primer entregable realizado. Se logró agilizar el proceso de control de documentación siendo realizado ahora por el sistema y no por el personal de Nutreco S.A. También los ingresos y egresos de planta mejoraron notablemente: Se simplificó el trabajo del personal de seguridad y de expedición, evitando así posibles errores.

Objetivo 2: Obtener una mejor información sobre los empleados de Nutreco y el cumplimiento del horario por parte de los mismos. Esto se logrará, primero cruzando los datos del sistema con los del reloj de la empresa para generar estadísticas e indicadores a partir de las discrepancias entre ambos registros/sistemas.

El cliente decidió que no se realice ningún control sobre sus empleados, por lo tanto, este objetivo fue modificado. En lugar de obtener información del horario de los empleados directos de la empresa se decidió realizar un seguimiento y control de los empleados tercerizados. De esta manera se agregó la posibilidad de crear un nuevo tipo de visitas: Visitas Externas. Las mismas cuentan con nombre completo y patente del vehículo del empleado tercerizado que realizará la visita, además de los horarios de llegada, salida, quien generó la visita, entre otros datos. Su creación quedó a cargo de una nueva entidad llamada Encargado de Visitas Externas.

Objetivo 3: Desarrollar una red neuronal que buscará la presencia de anomalías dentro de los horarios de ingreso y egreso de los empleados, en función de validar los

accesos registrados en el ingreso/egreso al predio y los ingresos/egresos registrados en el reloj del personal.

Dado que el objetivo anterior fue modificado, este tuvo que serlo también. En este caso se modificó para detectar anomalías dentro del horario de ingreso y egreso de los contratistas.

Este objetivo no pudo ser cumplido por solicitud del cliente. Durante el desarrollo del proyecto, aparecieron muchos requerimientos que eran prioritarios para él, por lo tanto decidió postergar el desarrollo de la red para etapas futuras. Esto se vio agravado por el hackeo que sufrieron en sus servidores, lo cual generó desconfianza e incertidumbre al momento de desarrollar la red neuronal.

Además, para entrenar la red neuronal se necesita un gran volumen de datos, por lo tanto, hasta que no se recopile esa información no puede comenzar su implementación. Cuando se retome este desarrollo en las siguientes etapas es muy probable que ya se cuente con la cantidad de datos necesarios.

Objetivo 4: Construir una red neuronal capaz de monitorear y validar los documentos previo a la auditoría del personal de la empresa, de manera que se simplifique el trabajo de los mismos. De esta forma, también se evita la carga de documentos incompletos y de mala calidad.

Como se mencionó en la sección 5.2.8, la idea era construir una red neuronal, pero, luego de investigar, se decidió utilizar otras técnicas de la inteligencia artificial para resolver la problemática. Cabe destacar que este objetivo, al igual que el anterior, también se vio afectado por el hackeo. No fue fácil lidiar con un cliente con tanta incertidumbre y desconfianza.

De todas formas, se podría decir que el objetivo fue parcialmente cumplido ya que el módulo desarrollado cumple la función planteada, pero no es exactamente lo mismo que lo que se planteó en un principio.

6.4 Desvíos detectados

Durante el desarrollo del proyecto hubo cambios en los requerimientos planteados inicialmente. Esto produjo contratiempos y modificaciones entre lo planeado desde un comienzo con lo finalmente construido.

6.4.1 Cambios en el flujo de ingreso

Flujo inicial: El conductor se presenta en la puerta con su código QR para que el personal de seguridad lo escanee desde la aplicación móvil. El usuario seguridad selecciona el vehículo con el cual iba a ingresar el conductor y la cantidad de pallets de entrada (si fuese necesario) para registrar el ingreso. Luego, de no haber ninguna irregularidad, se genera la visita y el conductor ingresa al establecimiento.

Este requerimiento fue validado por el cliente tanto en la fase de análisis como en la de diseño. Aun así, durante una demostración del primer entregable el cliente solicitó una modificación en este flujo, quedando de la siguiente manera:

Flujo modificado: El comienzo del flujo es igual al anterior, con el agregado de que, luego de que el usuario seguridad escanea y selecciona el vehículo y cantidad de pallets, el conductor no ingresa a la planta, sino que se lo agrega a una cola a la espera que el personal de expedición (nuevo actor) autorice su ingreso desde la web. Recién cuando se autoriza esto, el personal de seguridad marca su ingreso y se registra la visita.

Este cambio no solo generó un nuevo flujo de ingreso, sino que además se tuvo que agregar el nuevo rol de expedición con todo lo que ello conlleva: nueva entidad en la base de datos, nuevas pantallas en la web, nuevos *endpoints*, etc.

6.4.2 Control de empleados tercerizados

Como se menciona en la sección anterior, el cliente no estaba interesado en llevar un control del ingreso, egreso y documentación de sus empleados, en cambio si quería realizarlo para los empleados de mantenimiento. Para ello se agregó una categoría especial de visitas llamada “Visitas externas” y se agregó un nuevo rol llamado “Encargado de visitas externas”. Este sería quien se encargue de las altas, bajas y modificaciones de las visitas externas.

Este requerimiento se cambió antes de comenzar con su desarrollo, por lo tanto no produjo grandes contratiempos en el equipo.

6.4.3 Estado de los recursos

Otro cambio solicitado estuvo relacionado con el estado de un recurso. Esto era un indicador para saber si el recurso estaba apto o no para ingresar a la planta. Si la documentación estaba al día entonces el recurso estaba en estado “Valido”, de lo contrario su estado era “Invalido”. El cambio solicitado fue que este estado se pueda ver en la lista general de los recursos. Hasta ese momento se podía saber entrando a la pantalla detalle de cada uno. Allí se realizaba el mismo cálculo que se realizaba en el ingreso a planta para saber si ningún documento estaba vencido, pendiente o faltante. El hecho de mostrar este dato en la lista general implicaba un costo demasiado alto, ya que había que calcularlo para muchos recursos a la vez.

Luego de analizar el cambio se decidió implementar una solución desnormalizando el dato en la base. De está manera, cada recurso tendría una propiedad correspondiente a su estado. Esto implicó que debieran hacerse cálculos extra, ya que dicho atributo depende de la documentación asociada al recurso. Pero el costo de realizar estos cálculos era significativamente menor a las otras alternativas. Además, la desnormalización de estado significó ya tener acceso al mismo en el proceso de ingreso, evitando así su cálculo.

6.4.4 Código QR en el detalle de cada conductor

El código QR de cada conductor solo podía ser visualizado en la sección “Mi QR” ingresando con las credenciales del conductor. De esta forma el conductor podía exhibirlo en la entrada al establecimiento. Luego de una de las reuniones el cliente pidió que esta opción también se encuentre en los detalles del conductor para los roles de contratista y administrador.

Este cambio sólo afectó al *frontend*. Gracias a las buenas prácticas de programación y la reutilización de los componentes, no fue un cambio que haya demandado mucho tiempo y esfuerzo.

6.4.5 Modificación de la cantidad de pallets de salida luego de evaluar un arribo

El cliente solicitó que la cantidad de pallets de salida pueda ser modificable una vez que ya se evaluó un arribo. Esto se debió a que muchas veces se desconocía este número hasta que el conductor se marchaba de la planta.

El cambio no fue sencillo pero a su vez tampoco fue tan demandante. Se crearon los métodos de update y se actualizó el *frontend* para albergar un modal que permitiera ingresar la cantidad.

6.4.6 Modificación de responsabilidades del auditor

Tras una reunión interna en la empresa Nutreco se decidió que el rol Auditor también debería poder evaluar excepciones. De esta manera, los usuarios que se encargarían de las excepciones serían: Encargado, Auditor y Admin.

No fue un cambio demandante gracias a la arquitectura del sistema. Desde un principio se conocía el riesgo de sufrir modificaciones en las responsabilidades de los usuarios, por lo tanto, el *front-end* se pensó como componentes que podían ser compartidos por diferentes roles. Solamente se tuvo que agregar el componente al ruteo inicial del rol Auditor.

6.4.7 Acceder a QR de conductor desde la aplicación móvil

Otro de los desvíos que se detectaron en el proyecto fue la posibilidad de que cada conductor pueda acceder a su código QR desde la aplicación móvil. Para ello se tuvo que modificar el flujo de *login* de forma tal que acepte a los usuarios con rol “Conductor” y redirigirlos a una pantalla donde se muestre el código.

6.4.8 Hackeo a Nutreco

Este es sin dudas el desvío que más afectó al proyecto. Si bien no influyó particularmente en ningún objetivo en particular, produjo un gran contratiempo en su totalidad. Se dilató el tiempo en que se comenzó a utilizar el sistema y se tuvieron que volver a realizar tareas como el despliegue o la carga de datos. Además, la desconfianza

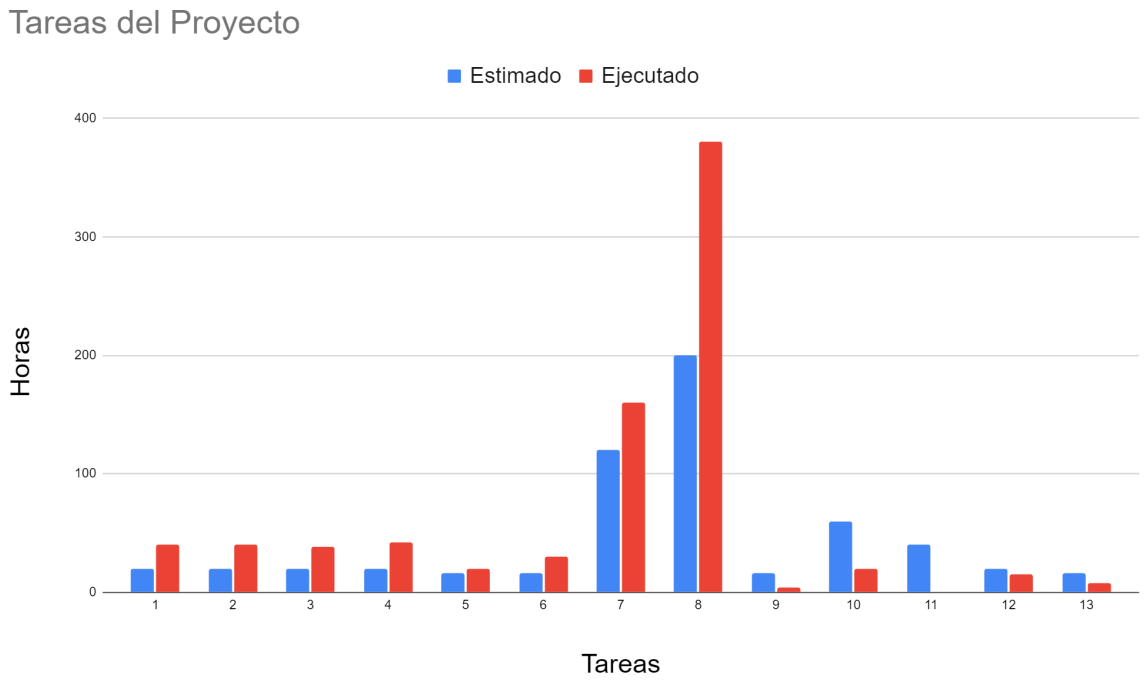
que se generó internamente en la empresa fue algo notable e impactó de forma muy negativa en el desarrollo del proyecto.

6.5 Planificación vs Ejecución

Luego de finalizado el proyecto, se realizó un análisis respecto al tiempo estimado con el tiempo con el objetivo de identificar donde se produjeron las las desviaciones más importantes y entender porqué el proyecto duró más de lo esperado.

Nº	**Tarea**	**Tiempo Estimado**	**Tiempo Ejecutado**	**Desviación**
1	Análisis de requerimientos del sistema	20 horas	40 horas	+20 horas
2	Relevamiento de Infraestructura de la empresa	20 horas	40 horas	+20 horas
3	Diseño de la Interfaz del sistema	20 horas	38 horas	+18 horas
4	Diseño de la Arquitectura del Sistema	20 horas	42 horas	+22 horas
5	Diseño de la base de datos	16 horas	20 horas	+4 horas
6	Armado del ambiente de desarrollo	16 horas	30 horas	+14 horas
7	Desarrollo frontend	120 horas	160 horas	+40 horas
8	Desarrollo backend	200 horas	380 horas	+180 horas
9	Desarrollo de base de datos	16 horas	4 horas	-12 horas
10	Desarrollo del monitor de documentos con alertas de auditoría	60 horas	20 horas	-40 horas
11	Desarrollo de la Red Neuronal de Anomalías	*(40 horas)	Por implementar	
12	Pruebas	20 horas	15 horas	-5 horas
13	Implementación del sistema	16 horas	8 horas	-8 horas
14	Documentación del proyecto	Durante todo el proyecto	Durante todo el proyecto	-
15	10% Overhead	60 horas	-	-
	Total	604 horas	797 horas	+253 horas

*Las 40 horas del desarrollo de la red neuronal de anomalías no fueron tenidas en cuenta en el total de horas ya que no fue implementada.



* La tarea N° 11 no se implementó

En base al análisis del gráfico de duración de las tareas, se han identificado varios factores que explican la desviación de 234 horas. En primer lugar, la inexperiencia del equipo en la realización de estimaciones para este tipo de proyectos ha sido un desafío significativo. El equipo carecía de práctica previa en proyectos reales, lo que limitó su capacidad para estimar con precisión la duración de las tareas. Además, la presencia de numerosas tareas desconocidas generó incertidumbre en cuanto a su dificultad y complejidad.

Por otro lado, las tareas de análisis, diseño y desarrollo se vieron seriamente afectadas por los obstáculos mencionados en la sección 6.2.1, así como por los cambios de requerimientos mencionados en la sección 6.4. Cada cambio requería un ciclo adicional de análisis, diseño, validación y desarrollo, lo que aumentó la duración de estas etapas. Especialmente los cambios surgidos durante el desarrollo del proyecto obligaron al equipo a modificar funcionalidades ya finalizadas para adaptarlas a los nuevos requerimientos, lo que generó un impacto adicional en la duración de las tareas.

La desviación más grande se puede ver en la tarea número 8, el desarrollo backend. Esto puede explicarse principalmente por el cambio en el flujo de ingreso a planta, lo cual tuvo un impacto directo en la duración de la tarea. Se invirtió tiempo en el desarrollo de una

versión del flujo que fue validada, pero posteriormente el cliente solicitó un cambio en dicho flujo. Además, la elección inicial de la tecnología de GraphQL, que luego fue reemplazada por REST, también contribuyó a la desviación en la duración de la tarea.

Además de esta desviación, se pueden destacar otras desviaciones significativas. Por ejemplo, la configuración de los pipelines generó contratiempos importantes en el armado del ambiente de desarrollo. Asimismo, el incidente de hackeo que sufrió el servidor de Nutreco tuvo como consecuencia la necesidad de implementar el sistema en dos ocasiones.

También puede verse que las últimas 5 tareas finalizaron antes de lo estimado inicialmente. Este hecho pone de manifiesto la falta de experiencia del equipo, ya que estas tareas fueron aprendidas durante la realización del proyecto, dificultando la precisión en la estimación de su dificultad. Entre ellas, se destaca la tarea número 13, referente a la implementación del sistema. Como se mencionó en el párrafo anterior, tuvo que ser realizada dos veces, pero aun así su duración fue la mitad de la estimada.

Cabe destacar que hubo muchos meses en los cuales no se realizaron trabajos debido a que el cliente, por motivos internos, no comenzaba con la utilización del sistema. Si bien no impactó directamente en el cálculo de horas, ya que fueron meses donde no se trabajó, si impacto en el tiempo de finalización del proyecto.

6.6 Trabajos futuros

Si bien el producto cumple con las expectativas del cliente, aún existen muchas funcionalidades y mejoras que se pueden agregar. Gracias a la arquitectura utilizada, la aplicación tiene mucho margen para escalar.

Hasta el día de la fecha, Nutreco solamente implementó el sistema en su planta ubicada en San Rafael. La idea es comenzar a utilizarlo en la planta de Sierras de los Padres en el corto plazo. Esto puede ser posible sin agregar o modificar una sola línea de código ya que el sistema fue pensado para poder soportar múltiples plantas a la vez.

El proceso de facturación con los proveedores transportistas fue uno de los trabajos que fueron postergados para el desarrollo de nuevos requerimientos. Por lo tanto, la idea es

implementarlo en el futuro ya que es una funcionalidad que la empresa necesitará en el mediano plazo. Permitirá una mayor transparencia en las transacciones, reducirá los errores y agilizará los pagos, lo que contribuirá a mejorar la eficiencia y productividad en el ámbito de la facturación con los proveedores transportistas en el futuro.

En cuanto a las redes neuronales, se propone implementar una para la detección de anomalías en el ingreso de contratistas. Antes de la implementación, se deberá recopilar los datos necesarios para entrenar la red y garantizar su efectividad. La red neuronal utilizará técnicas avanzadas de aprendizaje profundo y se realizarán pruebas y ajustes para mejorar su rendimiento continuamente. Esta propuesta busca garantizar la seguridad y proteger la integridad de la organización en relación con el ingreso de los contratistas.

El producto carece de diseño *UI/UX.* Al no contar con un diseñador en el equipo, todos los aspectos visuales fueron creados por desarrolladores. Un rediseño de las pantallas sería una gran mejora.

Actualmente la distribución de la aplicación se hace por archivo .apk al celular del usuario, ya que solo utilizan dispositivos android. Si en un futuro la aplicación escala a mayor cantidad de usuarios, subirla a *Play Store* y *App Store* no solo simplificará la distribución, sino que además se podrían generar grupos limitados para su testeo. Es por eso que a futuro se planea tener la aplicación móvil disponible en tiendas digitales.

7. Conclusiones

El proyecto fue largo pero satisfactorio. Durante el mismo hubo muchos aprendizajes, aciertos y errores. La primera enseñanza fue el trato con un cliente real. Si bien la comunicación fue con el personal del área de sistemas de Nutreco, no todos allí eran desarrolladores. Por lo tanto, se tuvo que traducir el lenguaje técnico en frases coloquiales para ser entendido por el equipo completo.. También se aprendió a interpretar problemáticas reales y generar soluciones posibles. Existe una gran diferencia entre lo que el cliente pide con lo que realmente necesita.

Trabajar en equipo es un desafío. Las metodologías ágiles, junto con el mantenimiento del código, permitieron la organización necesaria para elaborar un producto de calidad. Si bien los integrantes ya contaban con experiencia del trabajo en equipo como desarrolladores, ninguno había desempeñado otro rol previamente. Se aprendió a analizar, diseñar, dirigir, liderar, hablar con el cliente, entre otras habilidades. No obstante, un error que se cometió fue subestimar la administración. Al principio, parecía que iba a ser sencillo, por lo que se prescindió de algunas reuniones de SCRUM. A medida que avanzó el tiempo, esto generó descoordinaciones entre los miembros, lo que resultó en mayores tiempos de entrega.

Se incorporó mucho contenido técnico. La solución planteada contaba con funcionalidades nunca antes utilizadas por el equipo. Entre ellas: Manejo de diferentes roles, generación y lectura de códigos QR, carga, descarga y almacenamiento de archivos. Se tuvieron que realizar investigaciones como, por ejemplo, búsqueda en comunidades, lectura de documentación de las diferentes librerías y frameworks, mirar cursos o *webinars* en línea, entre otras.

La elección de las tecnologías a utilizar fue un gran acierto por parte del equipo. No se tuvo que realizar cambios desde que se definieron cuales serían de utilidad para la resolución del problema, a excepción de una: *GraphQL.* Optar por esta solución fue un error. La decisión se basó en que brinda un mejor rendimiento respecto a REST. Pero, los problemas comenzaron a medida que el equipo se adentró en ella. Al ser relativamente nueva, se encontró poca documentación o información al respecto. Además, de que si bien desde la perspectiva del *FrontEnd* proporciona grandes alternativas y soluciones, la complejidad de implementación es mayor en el *BackEnd*. Eventualmente, se llegó a la conclusión que el desarrollo se estaba demorando más de lo estimado debido a la curva de

aprendizaje de está tecnología. Fue por eso que se decidió cambiar. Esto significó tener que dedicar tiempo a realizar estos cambios, pero agilizó considerablemente el desarrollo posterior.

Por otro lado, la estimación de los tiempos fue muy optimista. El proyecto duró más de lo previsto. Si bien los cambios en los requerimientos y los inconvenientes que surgieron a raíz del hackeo fueron de los factores más importantes, también hubo otros: Vacaciones, enfermedades, dificultad para reunir a todos los miembros involucrados (tanto del equipo como del cliente), entre otros.

Aun así, el equipo está muy contento con el proyecto realizado, tanto por los aciertos como por los errores. Estos últimos traen consigo una valiosa enseñanza que será muy útil para el día de mañana. Además, el cliente pidió continuar con la implementación del sistema, lo cual es un gran indicio de que la solución propuesta y el desarrollo resolvieron la problemática satisfactoriamente.

8. Referencias

[1] Santander Universidades. (2020, 21 de diciembre). *Metodologías de desarrollo de software: ¿qué son?*. https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html#:~:text=Ponerse%20a%20desarrollar%20un%20producto,definitiva%2C%20un%20mal%20resultado%20final.

[2] Ing. Suárez, Karina Mero. *Diseño y Desarrollo del Software*. Consultado el 24 de agosto de 2022. https://blogereducativo.wordpress.com/diseno-y-desarrollo-del-software/

[3] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, 2nd Edition, Addison Wesley, 2003.

[4] Logan Dev. (2023, 17 de enero). *The Most Demanded Frontend Frameworks in 2022.* https://www.devjobsscanner.com/blog/the-most-demanded-frontend-frameworks-in-2022/

Anexo I: Definiciones

Metodologías de trabajo

Kanban

Esta metodología se basa en el desarrollo y entrega continuos, abordando un pequeño número de tareas de forma fluida y simultánea. Se busca priorizar las tareas en curso frente a las nuevas. Los equipos Kanban utilizan una herramienta de planificación visual, el tablero Kanban, que muestra cada proyecto (historia del usuario) en una tarjeta y mueve esas tarjetas a través de columnas que representan etapas progresivas de finalización. Si el proyecto tiene un flujo continuo de solicitudes de trabajo, el Kanban puede ser el adecuado para gestionarlo. Por ejemplo, muchos equipos de soporte y mantenimiento usan Kanban en vez de Scrum, ya que con este último la tarea de gestión de su trabajo se hace casi imposible.

Scrum

Scrum también divide tareas complejas en historias de usuarios y las visualiza en un flujo de trabajo (muchas veces en un tablero Kanban). Los equipos de Scrum se comprometen a generar software al final de los intervalos establecidos, Sprints. Si tu necesidad es entregar valor a los clientes de forma regular, scrum parece el mejor camino.

Es un framework que se utiliza dentro de equipos que manejan proyectos de alta incertidumbre. Se trata de un marco de trabajo por el cual las personas pueden abordar problemas complejos adaptativos, a la vez que entregar productos del máximo valor posible productiva y creativamente. Scrum es liviano y fácil de entender pero, a la vez, difícil de dominar del todo. Este framework favorece el time to market y la entrega rápida de MVP (mínimos productos viables).

Las características de esta metodología son:

- Flexibilidad ante cambios y nuevos requisitos.
- Factor humano
- Colaboración e interacción con el cliente

- Desarrollo iterativo e incremental.

Arquitectura de software

La arquitectura de Software hace referencia a la estructura y la relación entre las diferentes partes de un software y sus propiedades visibles externas. Su principal objetivo radica en ofrecer cierta calidad al sistema de administración de datos, a partir de su desempeño, ahorro de tiempo, su disponibilidad y usabilidad, la capacidad de modificarse y adecuarse a las nuevas necesidades del sistema, entre otros atributos de calidad.

Frontend

El desarrollo *frontend* es aquel que da una estructura a los datos que aparecen en una interfaz gráfica, con el objetivo de optimizar la experiencia del usuario. Básicamente es quien permite al sistema interactuar con el usuario. Aquí juegan un rol muy importante los diseños visuales, ya sean colores, márgenes, animaciones, etc.

Framework: Es una estructura previamente elaborada que se puede aprovechar para desarrollar un proyecto. Es una especie de plantilla o esquema conceptual que simplifica la elaboración de una tarea, ya que solo es necesario complementarlo de acuerdo a lo que se quiere realizar. La gran ventaja es que nos garantizan el uso de buenas prácticas y ya están testeados para asegurarnos el correcto funcionamiento del código.

Progressive Web Apps (PWA): Es un sitio web con una interfaz y funcionalidades similares a las de una aplicación móvil. Se basan en el uso de Service Workers. Estos son procesos que se encargan de guardar recursos como archivos HTML o imágenes para aumentar la velocidad de carga del sitio y que la misma no se vea afectada por las condiciones de internet.

Aplicaciones Nativas: Son aplicaciones desarrolladas en el lenguaje de programación nativo de un sistema operativo específico. Las mismas deben ser instaladas desde la tienda de cada sistema operativo (App Store, Google Play Store).

Archivos .ipa: el formato utilizado para las aplicaciones de Apple en los dispositivos iPhone, iPod Touch y iPad.

Backend

El *backend* es la parte del desarrollo web que se encarga de que toda la lógica de una página web funcione. Se trata del conjunto de acciones que pasan en una web pero que no vemos como, por ejemplo, la comunicación con el servidor. No está visible a ojos del usuario y no incluye ningún tipo de elemento gráfico.

Arquitectura de microservicios

Con una arquitectura de microservicios, una aplicación se crea con componentes independientes que ejecutan cada proceso de la aplicación como un servicio. Estos servicios se comunican a través de una interfaz bien definida mediante API ligeras. Cada uno se puede desarrollar, implementar, operar y escalar sin afectar el funcionamiento de otros servicios.

Ventajas:

- Agilidad: Los microservicios fomentan una organización de equipos pequeños e independientes que se apropian de los servicios. Esto acorta los tiempos del ciclo de desarrollo ya que los equipos actúan en un contexto pequeño y bien definido.
- Escalado flexible: Los microservicios permiten que cada servicio se escale de forma independiente para satisfacer la demanda de la característica de la aplicación que respalda.
- Implementación sencilla: Los microservicios permiten la integración y la entrega continuas, lo que facilita probar nuevas ideas y revertirlas si algo no funciona. El bajo costo de los errores permite experimentar, facilita la actualización del código y acelera el tiempo de comercialización de las nuevas características.
- Libertad tecnológica: Las arquitecturas de microservicios no siguen un enfoque de "diseño único". Los equipos tienen la libertad de elegir la mejor herramienta para resolver sus problemas específicos. Como consecuencia, los equipos que crean microservicios pueden elegir la mejor herramienta para cada trabajo.
- Código reutilizable: La división del software en módulos pequeños y bien definidos les permite a los equipos usar funciones para diferentes propósitos. Un servicio escrito para una determinada función se puede usar como un componente básico para otra característica.
- Resistencia a errores: Un error en un servicio puede manejarse de forma que no bloquee toda la aplicación. En cambio, en una arquitectura monolítica, un error en un solo componente puede provocar un error en toda la aplicación.

Base de datos

Es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. Son el producto de la necesidad humana de almacenar la información, es decir, de preservarla contra el tiempo y el deterioro, para poder acudir a ella posteriormente. En ese sentido, la aparición de la electrónica y la computación brindó el elemento digital indispensable para almacenar enormes cantidades de datos en espacios físicos limitados, gracias a su conversión en señales eléctricas o magnéticas.

Servidor

Es un aparato informático que almacena, distribuye y suministra información. Los servidores funcionan basándose en el modelo “cliente-servidor”. El cliente puede ser tanto un ordenador como una aplicación que requiere información del servidor para funcionar. Por tanto, un servidor ofrecerá la información demandada por el cliente siempre y cuando el cliente esté autorizado. Los servidores pueden ser físicos o virtuales.

HTTP

Es el protocolo de comunicación que permite las transferencias de información a través de archivos en la World Wide Web. Es el código que se establece para que el computador solicitante y el que contiene la información solicitada puedan “hablar” un mismo idioma a la hora de transmitir información por la red.

API

Conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas.

REST

REST (REpresentational State Transfer) es un estilo arquitectónico para desarrollar servicios web. REST es popular debido a su simplicidad y al hecho de que se basa en los sistemas y características existentes del Protocolo de transferencia de hipertexto (HTTP) de

Internet para lograr sus objetivos, en lugar de crear nuevos estándares, marcos y tecnologías.

GraphQL

Es un lenguaje de consulta y un tiempo de ejecución del servidor para las interfaces de programación de aplicaciones (API); su función es brindar a los clientes exactamente los datos que solicitan y nada más.

Inteligencia Artificial

La inteligencia artificial es un campo de la ciencia relacionado con la creación de computadoras y máquinas que pueden razonar, aprender y actuar de una manera que normalmente requeriría inteligencia humana o que involucre datos cuya escala exceda lo que los humanos pueden analizar. Abarca muchas disciplinas diferentes, incluidas la informática, el análisis de datos y las estadísticas, la ingeniería de hardware y software, la lingüística, la neurociencia y hasta la filosofía y la psicología.

Anexo II: Diagramas

Análisis

Diagrama de contexto

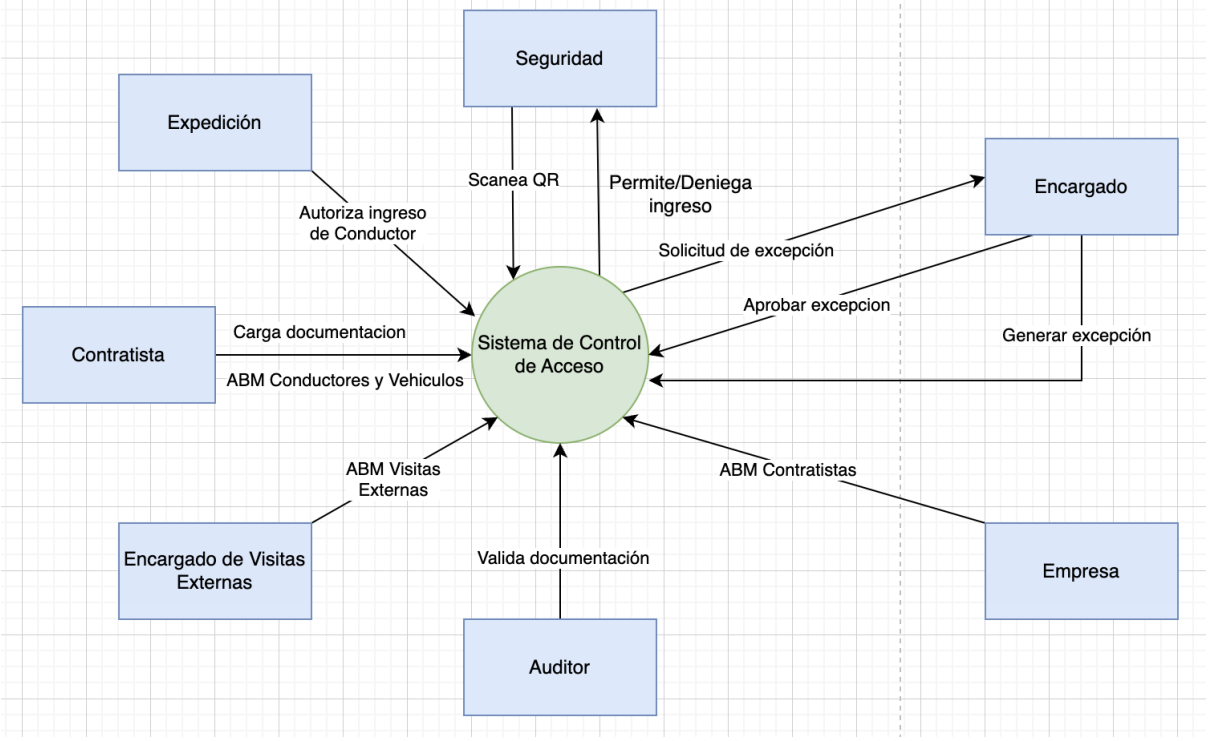


Diagrama de componentes

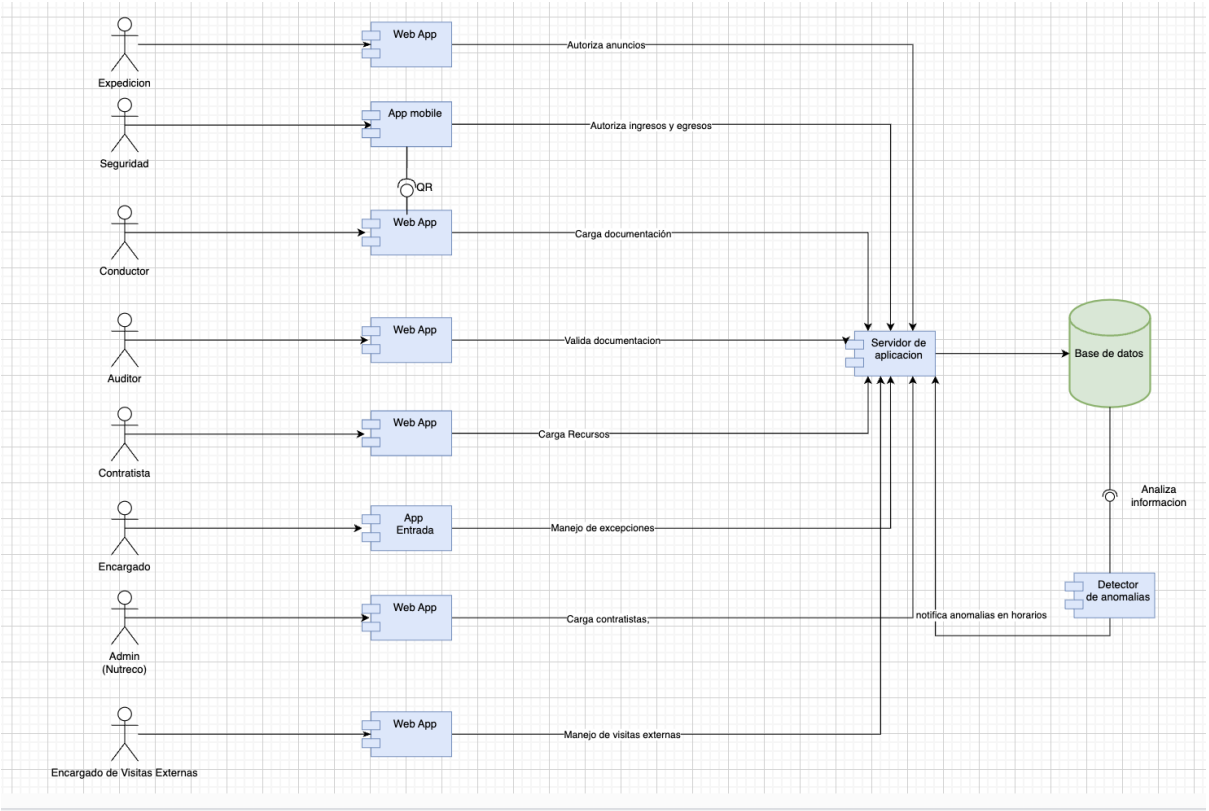
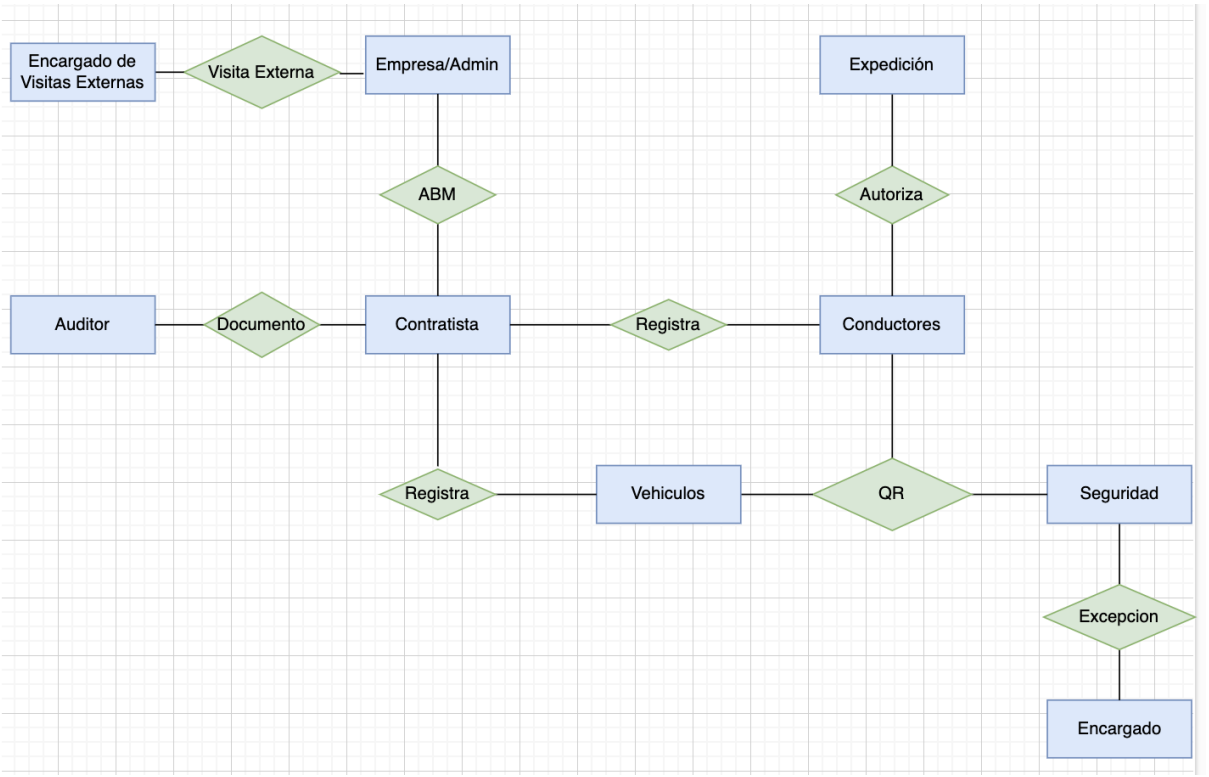
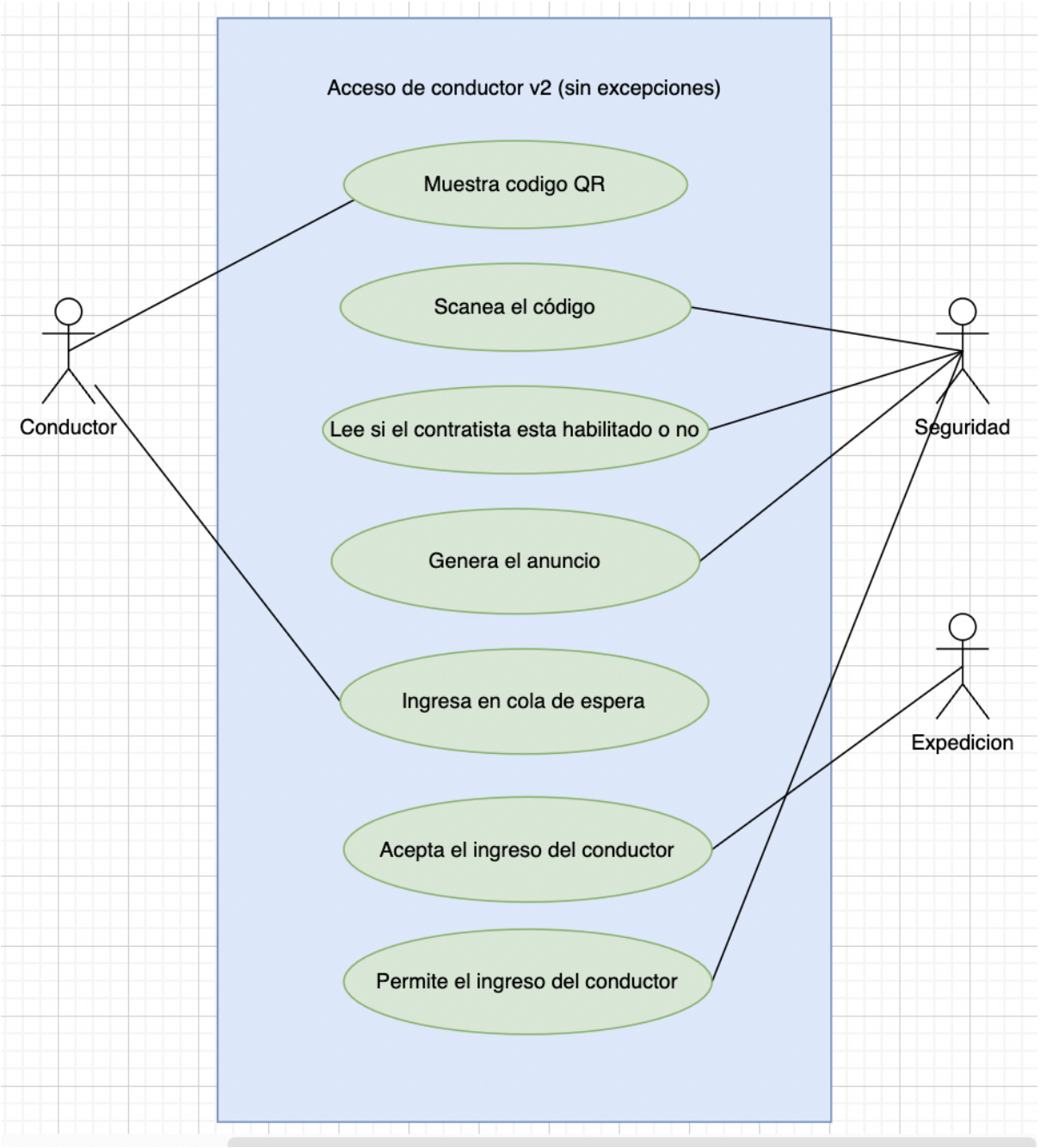
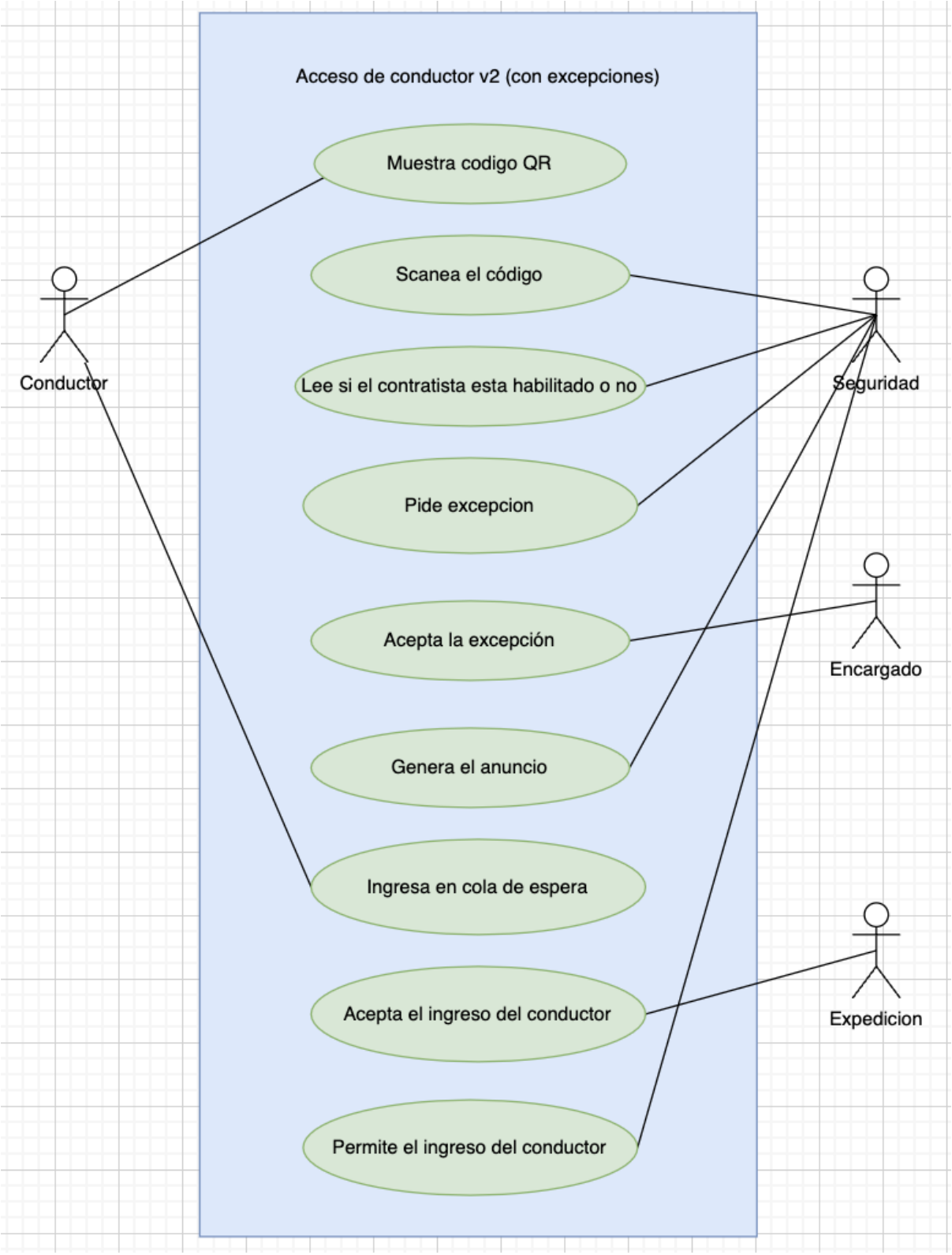


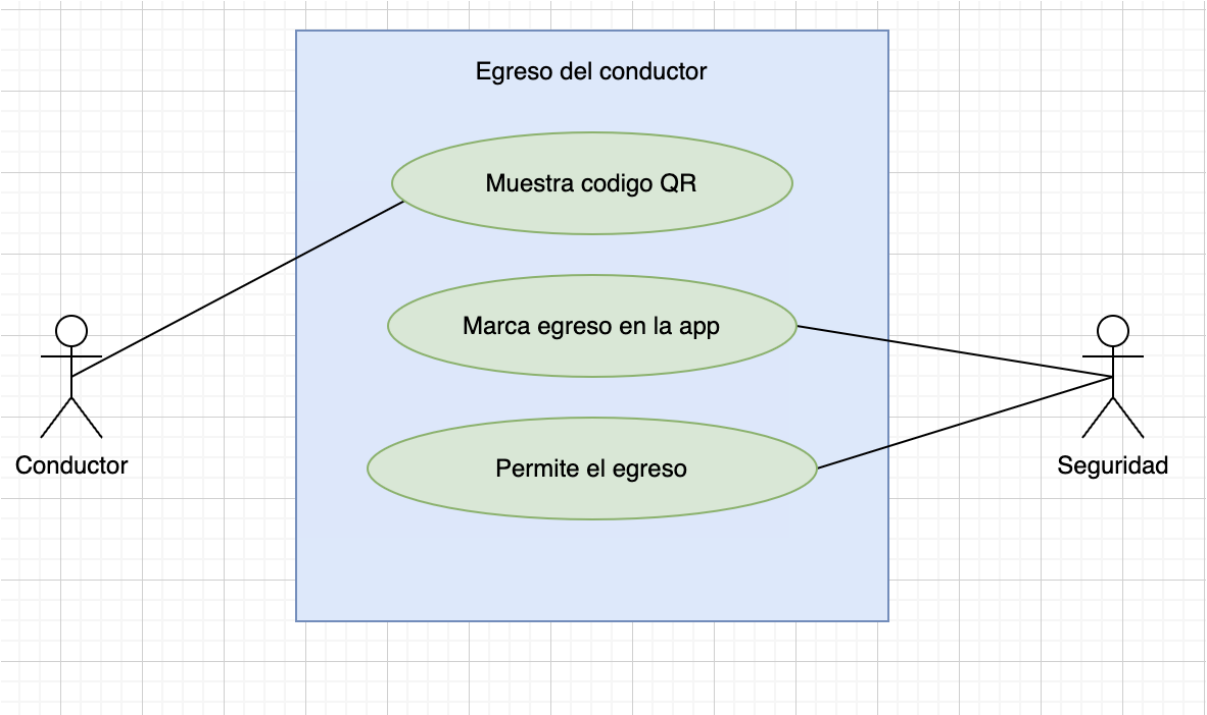
Diagrama de entidad-relación

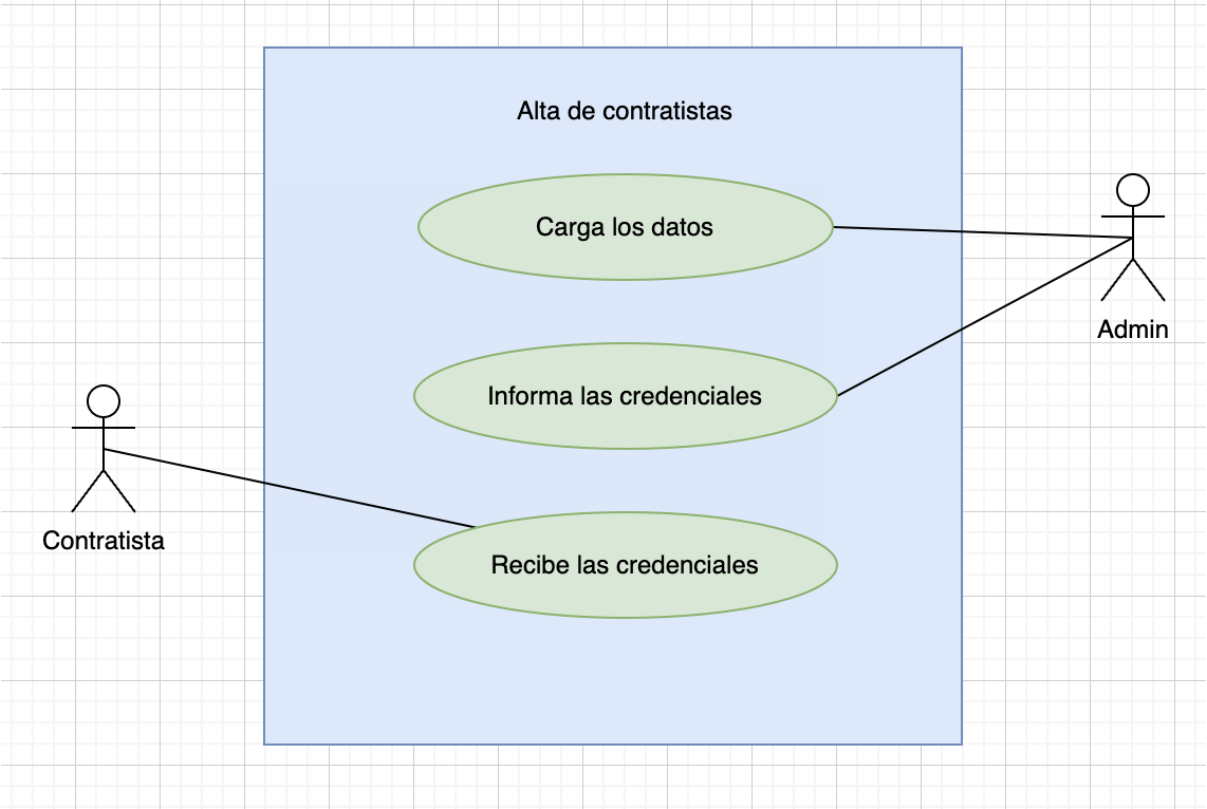


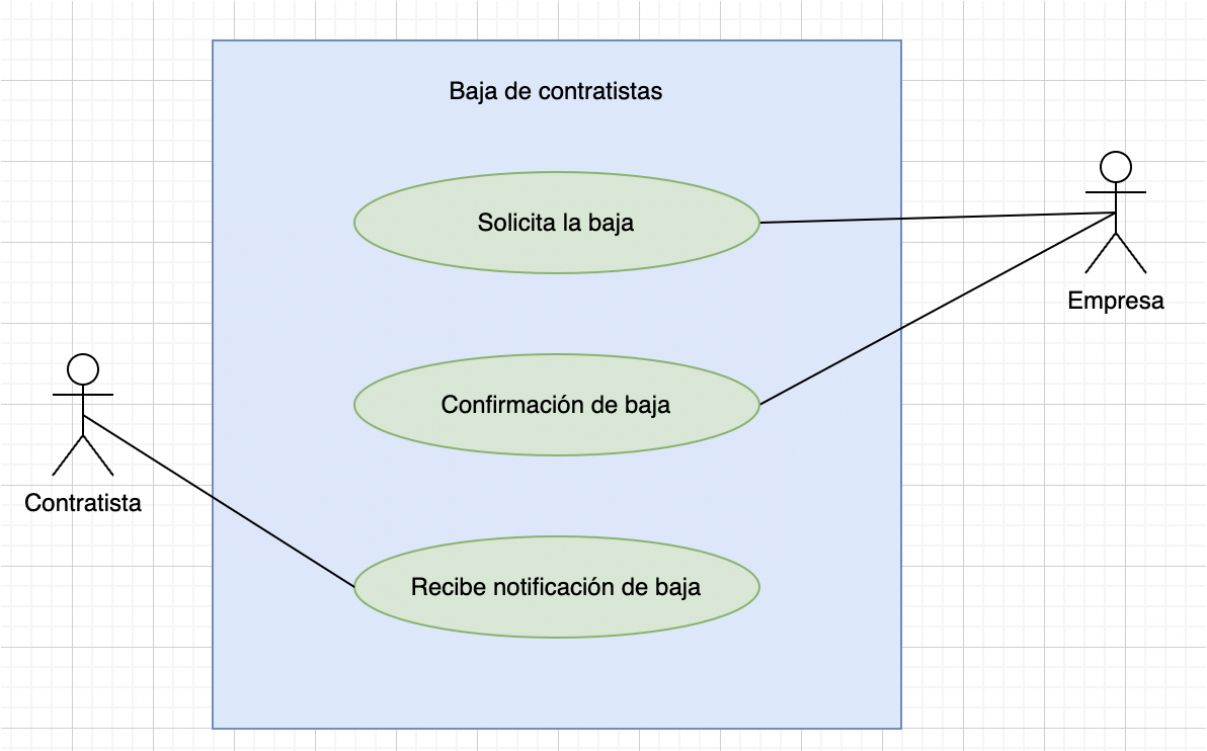
Diagramas de casos de uso




Acceso de conductor v2 (con excepciones)
Muestra codigo QR
Scanea el código
Lee si el contratista esta habilitado o no
Pide excepcion
Acepta la excepción
Genera el anuncio
Ingresa en cola de espera
Acepta el ingreso del conductor
Permite el ingreso del conductor
Conductor
Seguridad
Encargado
Expedicion

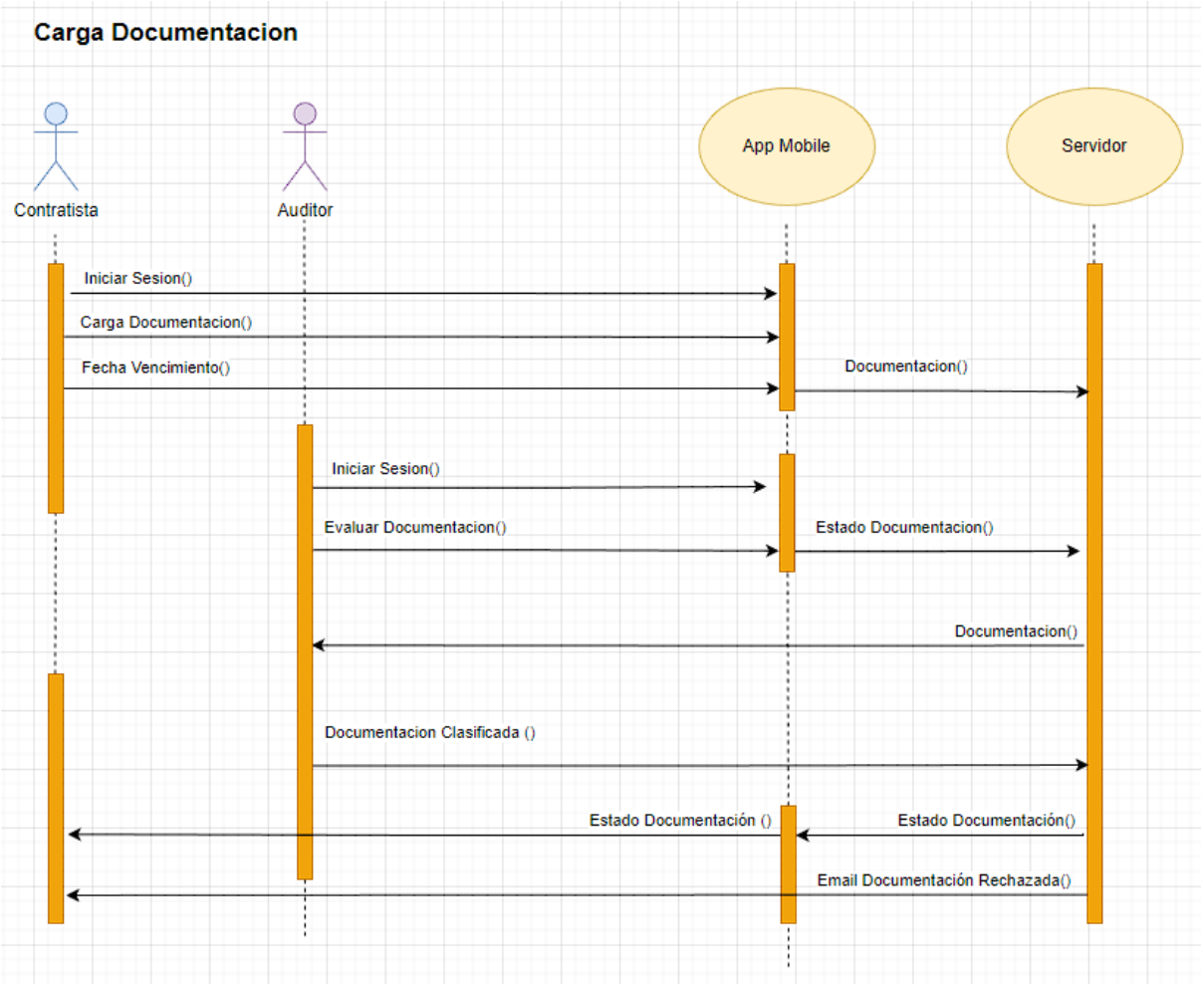

Egreso del conductor
Muestra codigo QR
Marca egreso en la app
Permite el egreso
Conductor
Seguridad

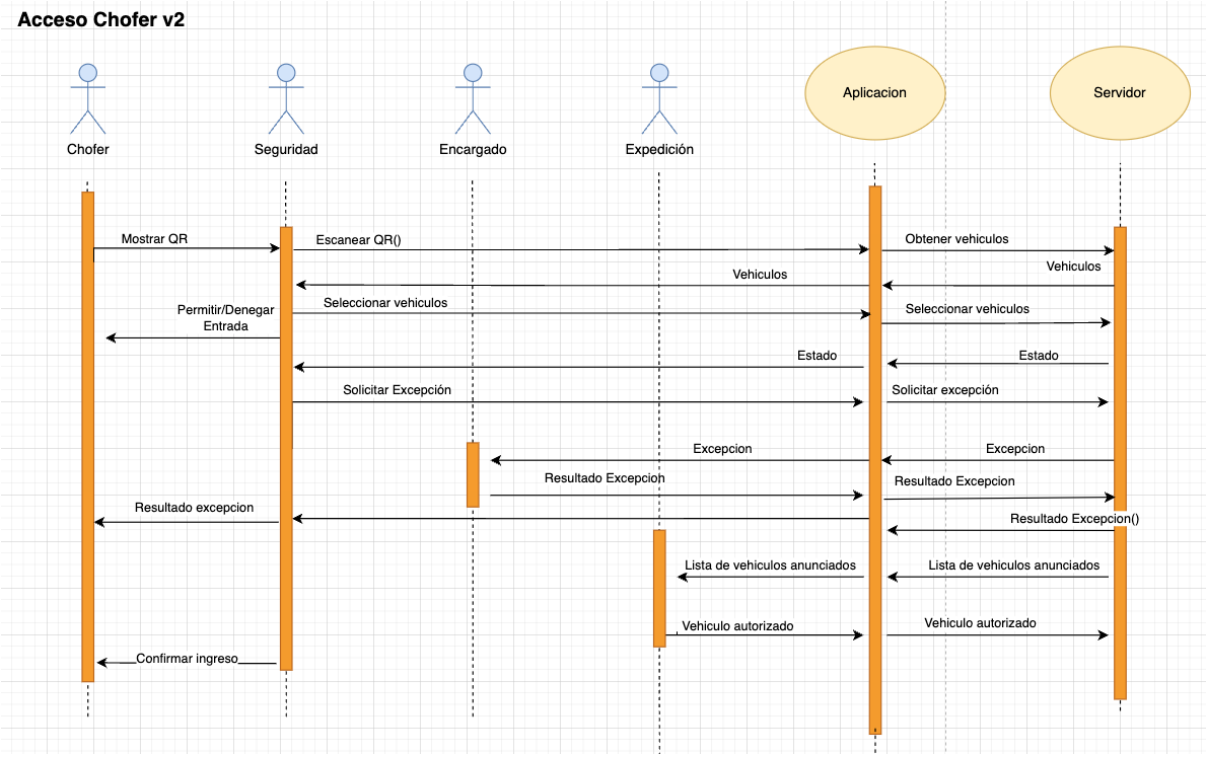

Alta de contratistas
Carga los datos
Informa las credenciales
Recibe las credenciales
Admin
Contratista

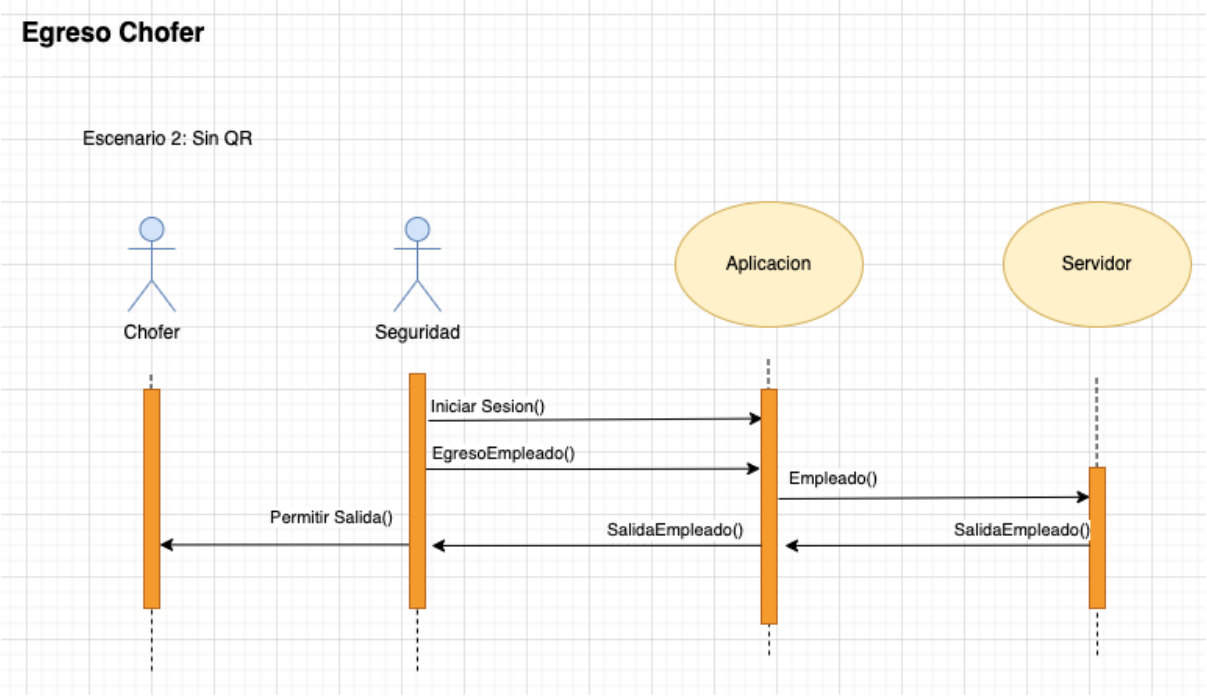

Baja de contratistas
Solicita la baja
Confirmación de baja
Recibe notificación de baja
Empresa
Contratista

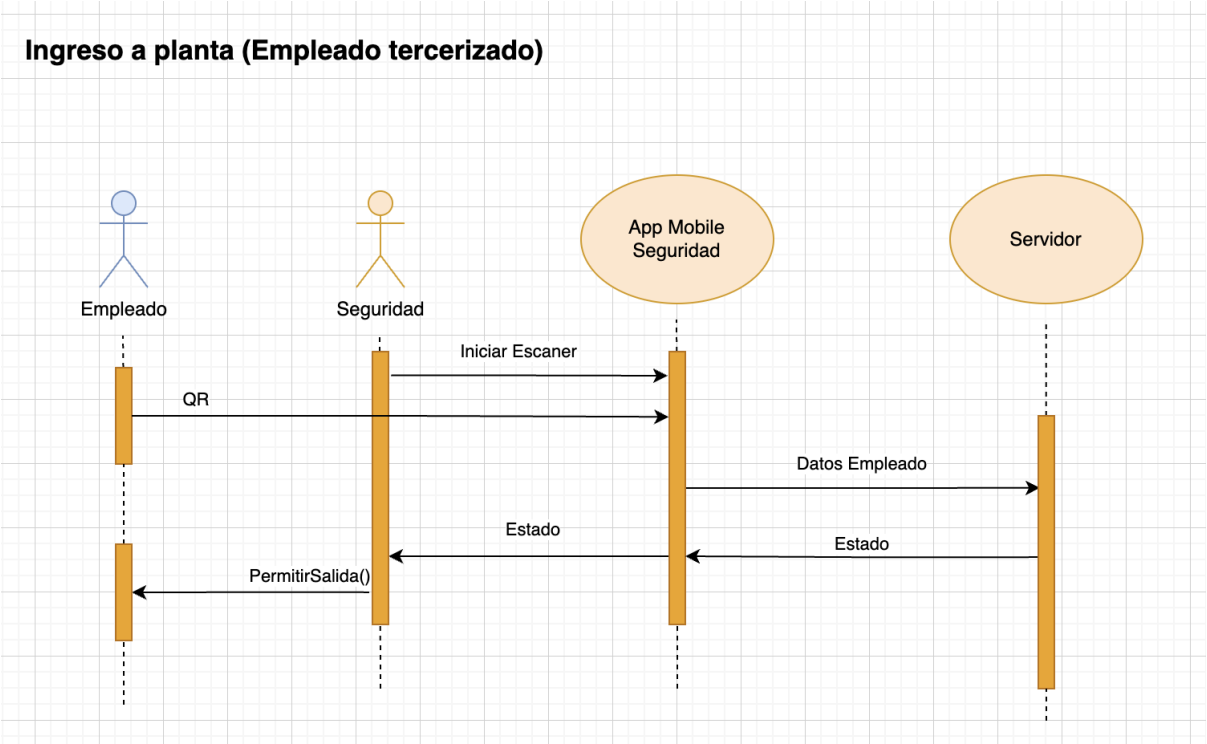

Carga/Modificacion de documentacion
Sube los archivos
Indicar fecha de vencimiento
Verifica la documentacion
Acepta/Rechaza documento
Lee los motivos por email (solo si se rechaza)
Contratista
Auditor

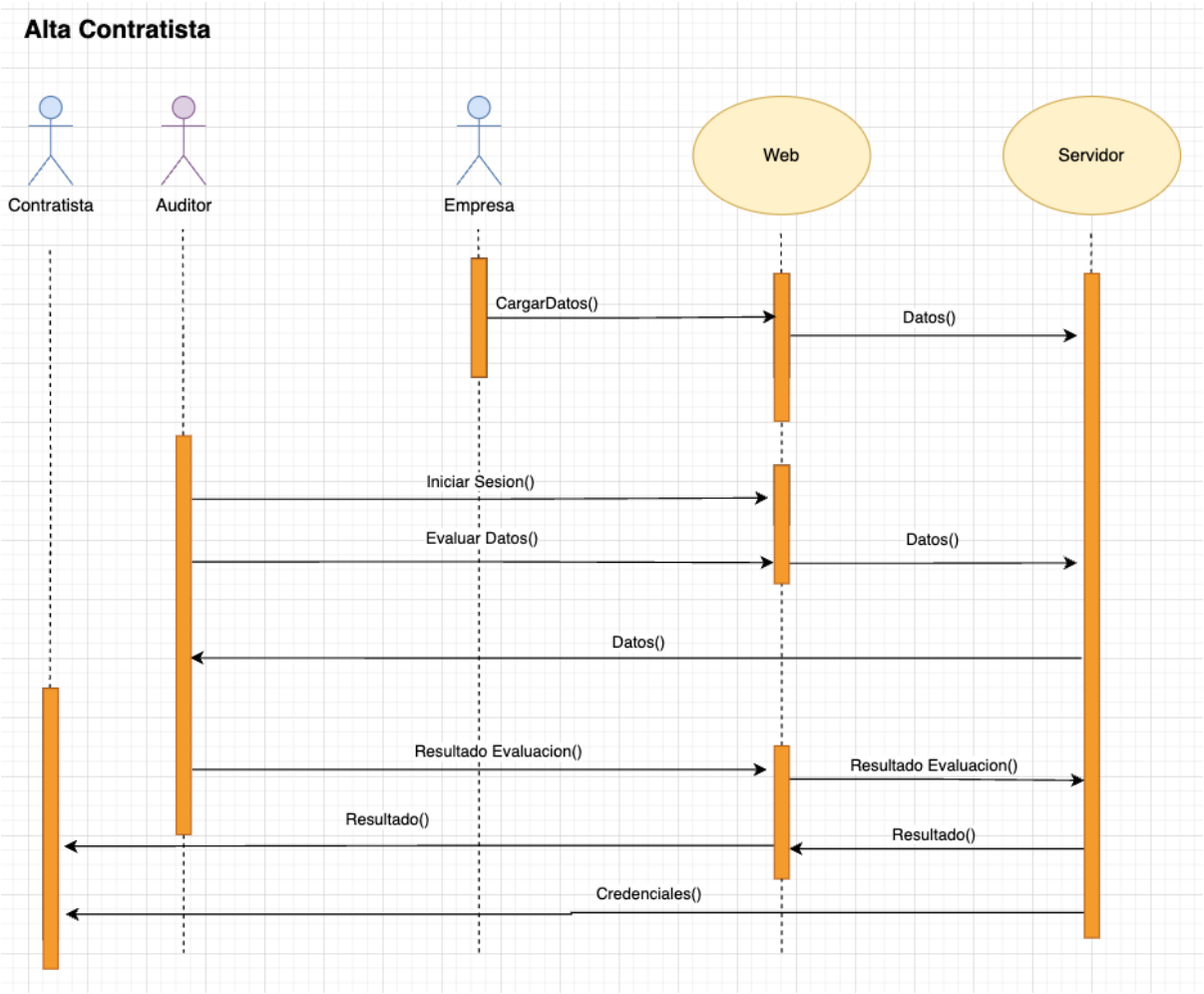
10.1.6 Diagramas de secuencia

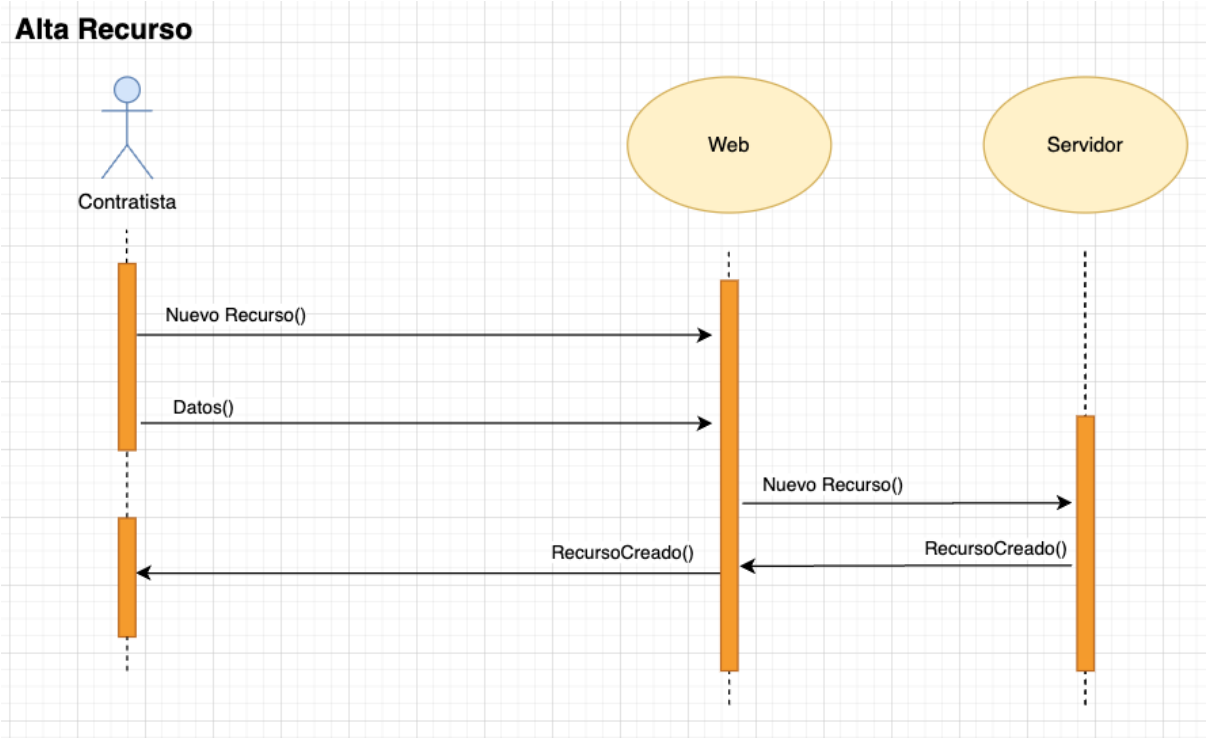


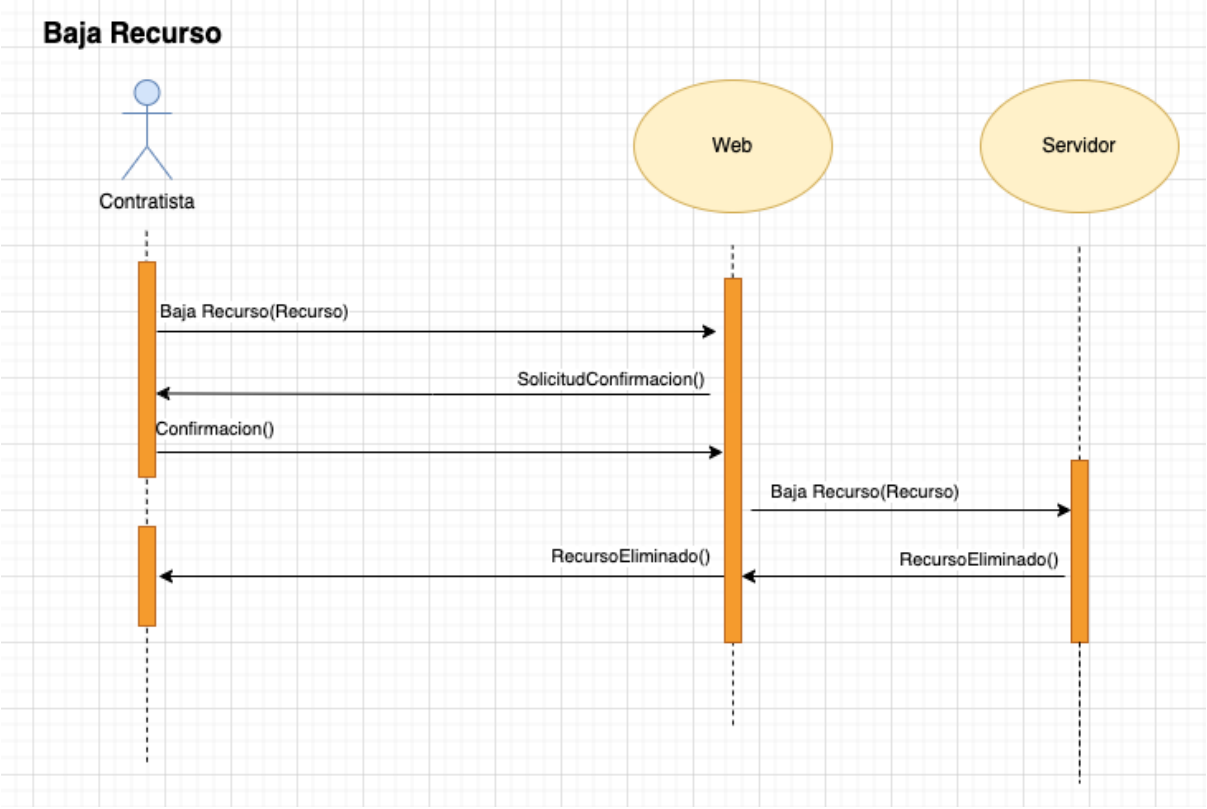

Acceso Chofer v2
Chofer
Seguridad
Encargado
Expedición
Aplicacion
Servidor
Mostrar QR
Escanear QR()
Obtener vehiculos
Vehiculos
Vehiculos
Permitir/Denegar Entrada
Seleccionar vehiculos
Seleccionar vehiculos
Estado
Estado
Solicitar Excepción
Solicitar excepción
Excepcion
Excepcion
Resultado Excepcion
Resultado Excepcion
Resultado excepcion
Resultado Excepcion()
Lista de vehiculos anunciados
Lista de vehiculos anunciados
Vehiculo autorizado
Vehiculo autorizado
Confirmar ingreso

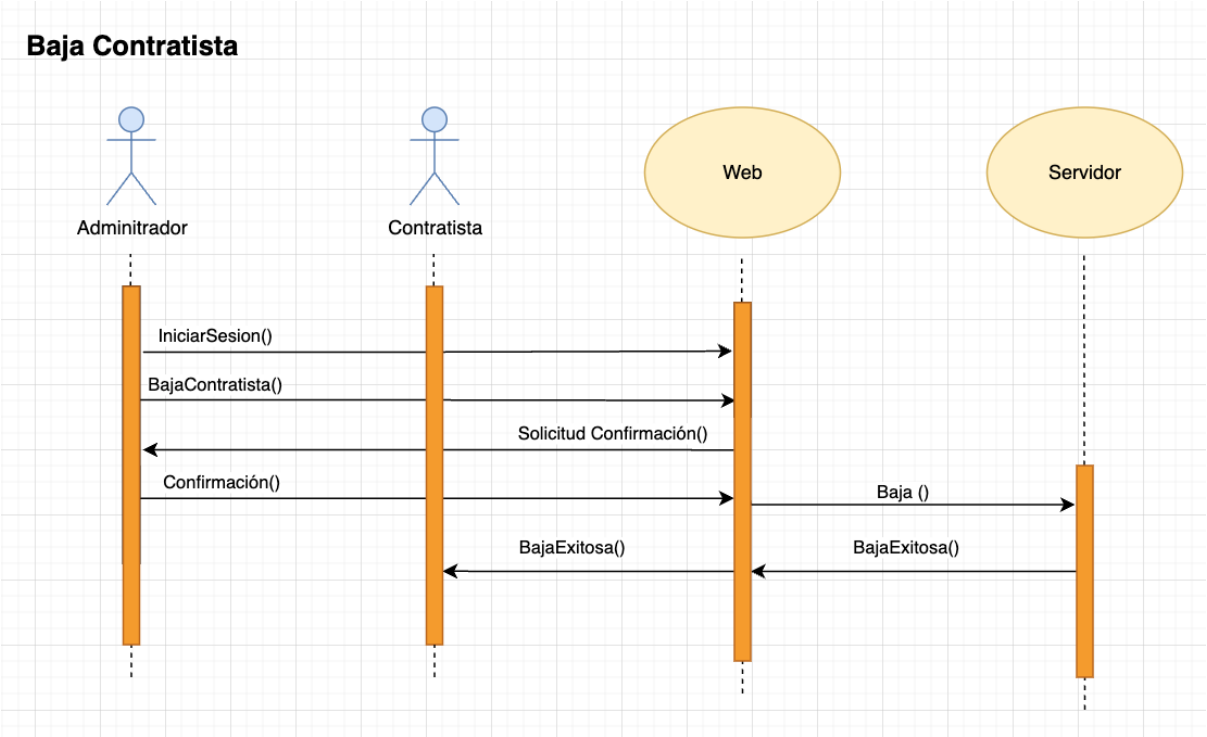

Egreso Chofer
Escenario 2: Sin QR
Chofer
Seguridad
Aplicacion
Servidor
Iniciar Sesion()
EgresoEmpleado()
Empleado()
Permitir Salida()
SalidaEmpleado()
SalidaEmpleado()

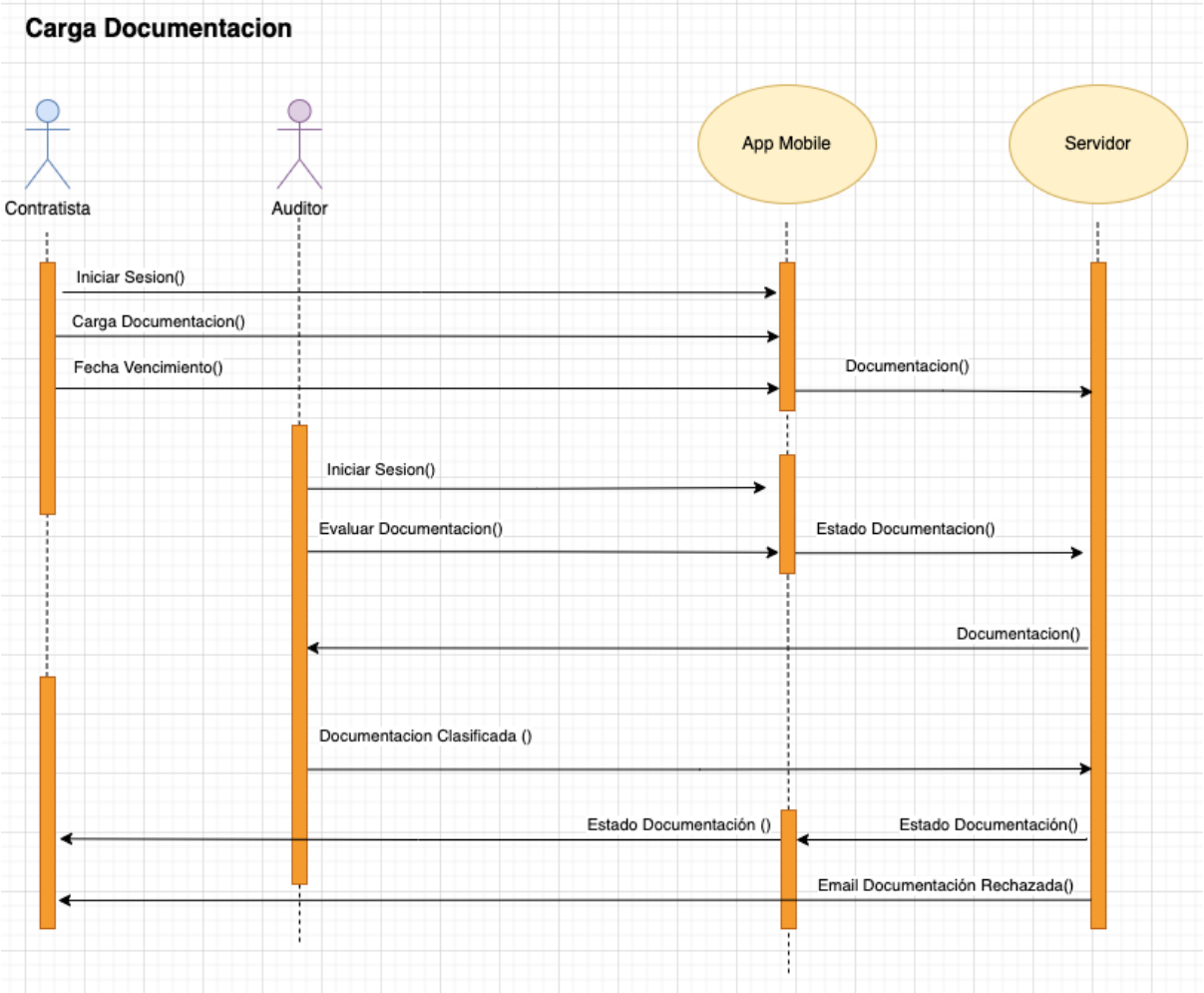

Ingreso a planta (Empleado tercerizado)
Empleado
Seguridad
App Mobile Seguridad
Servidor
Iniciar Escaner
QR
Datos Empleado
Estado
Estado
PermitirSalida()

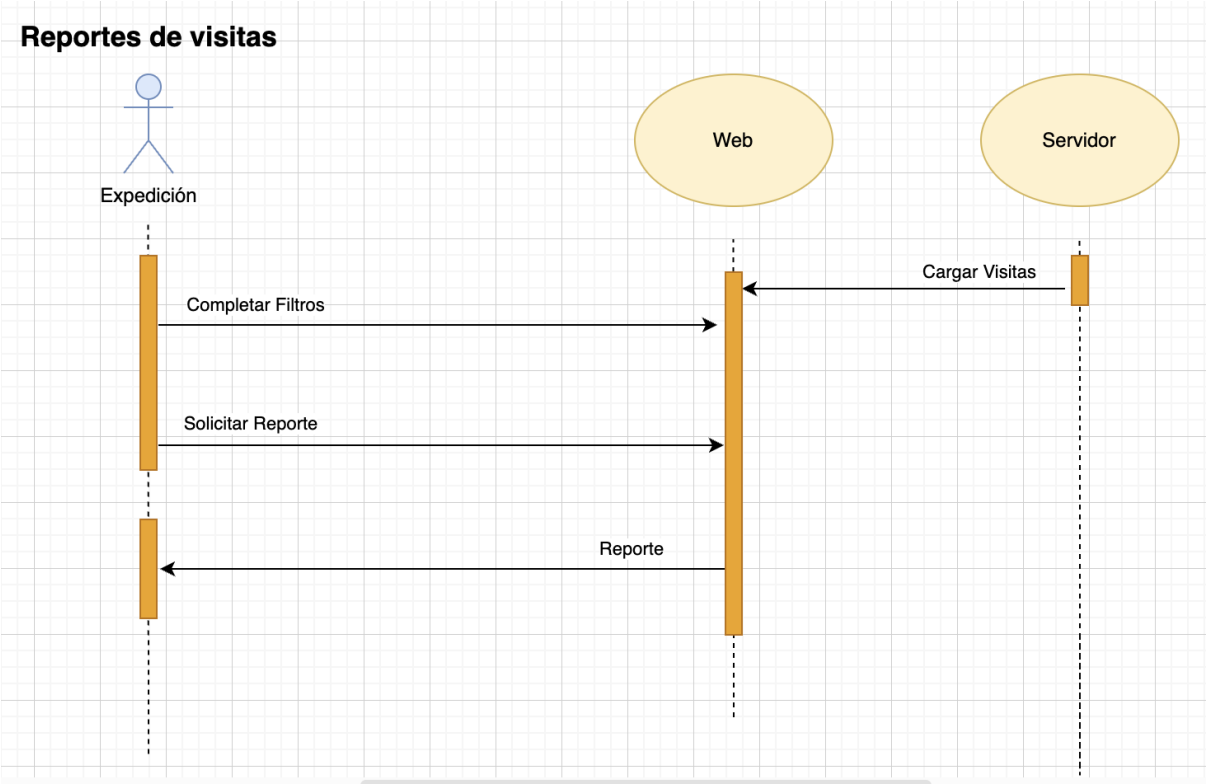

Alta Contratista
Contratista
Auditor
Empresa
Web
Servidor
CargarDatos()
Datos()
Iniciar Sesion()
Evaluar Datos()
Datos()
Datos()
Resultado Evaluacion()
Resultado Evaluacion()
Resultado()
Resultado()
Credenciales()


Alta Recurso
Contratista
Web
Servidor
Nuevo Recurso()
Datos()
Nuevo Recurso()
RecursoCreado()
RecursoCreado()

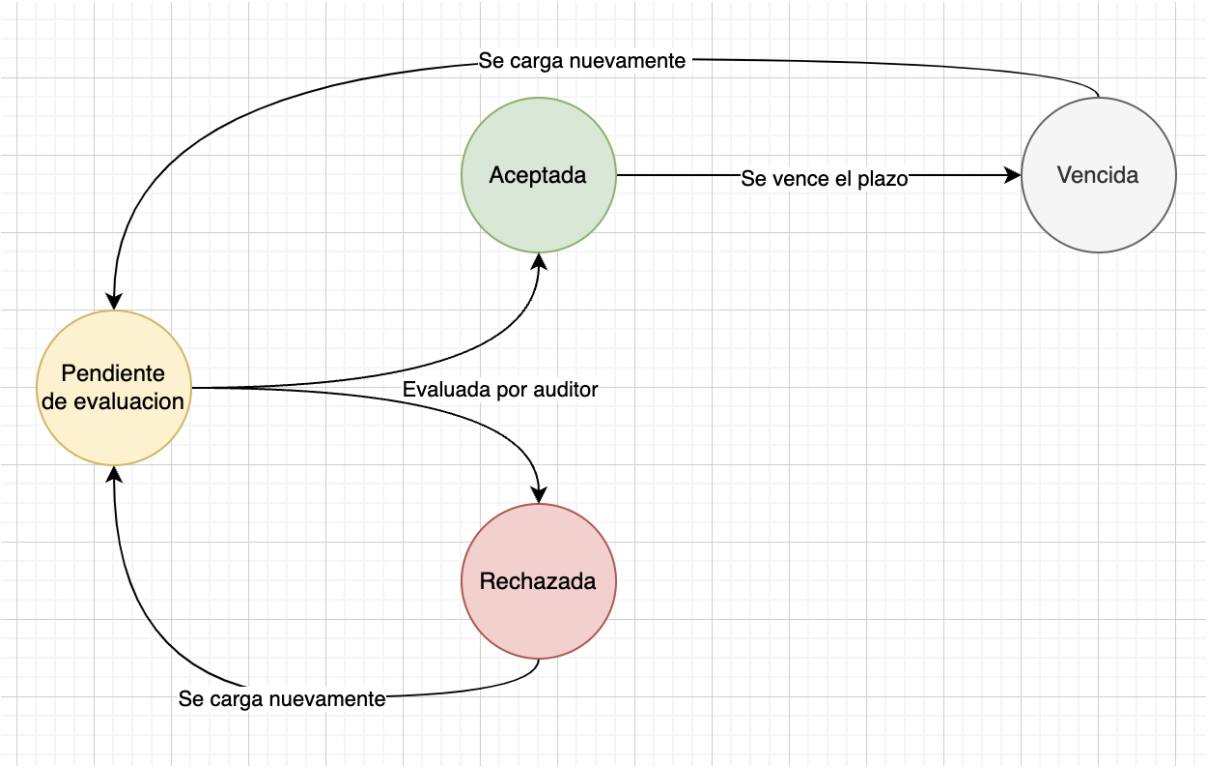

Baja Recurso
Contratista
Web
Servidor
Baja Recurso(Recurso)
SolicitudConfirmacion()
Confirmacion()
Baja Recurso(Recurso)
RecursoEliminado()
RecursoEliminado()


Baja Contratista
Adminitrador
Contratista
Web
Servidor
IniciarSesion()
BajaContratista()
Solicitud Confirmación()
Confirmación()
Baja ()
BajaExitosa()
BajaExitosa()


Carga Documentacion
Contratista
Auditor
App Mobile
Servidor
Iniciar Sesion()
Carga Documentacion()
Fecha Vencimiento()
Documentacion()
Iniciar Sesion()
Evaluar Documentacion()
Estado Documentacion()
Documentacion()
Documentacion Clasificada ()
Estado Documentación ()
Estado Documentación()
Email Documentación Rechazada()

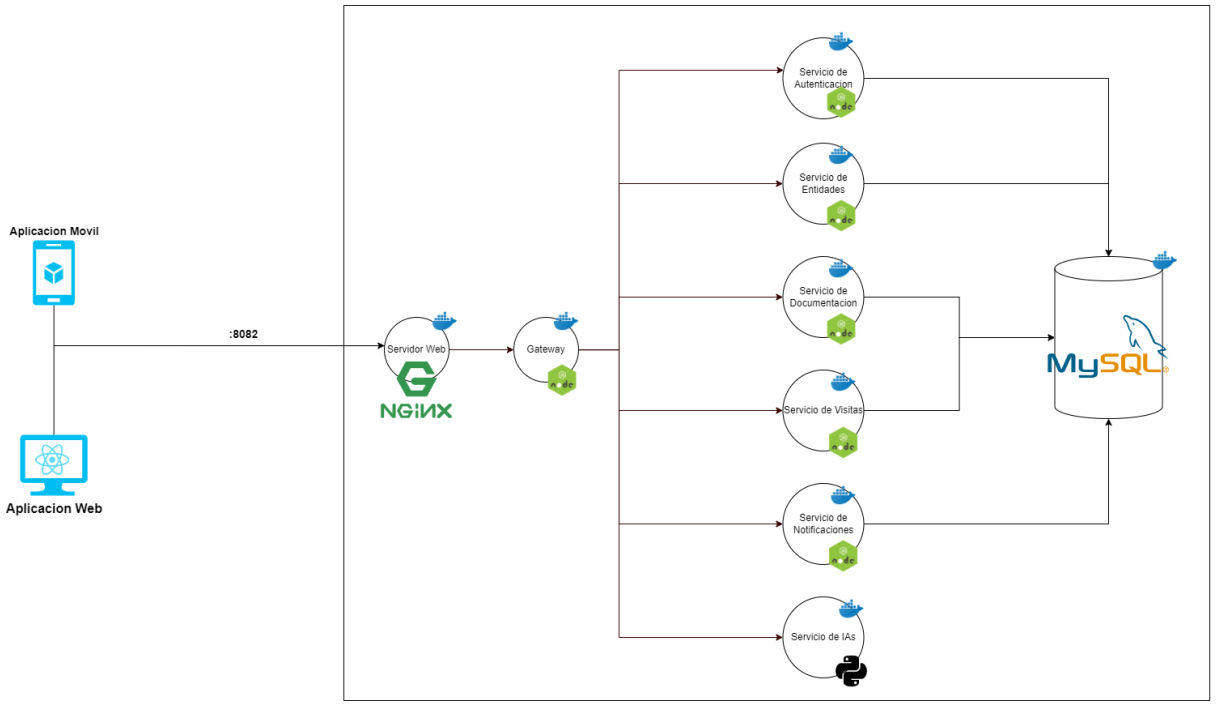


10.1.6 Diagrama de transición de estados de los documentos



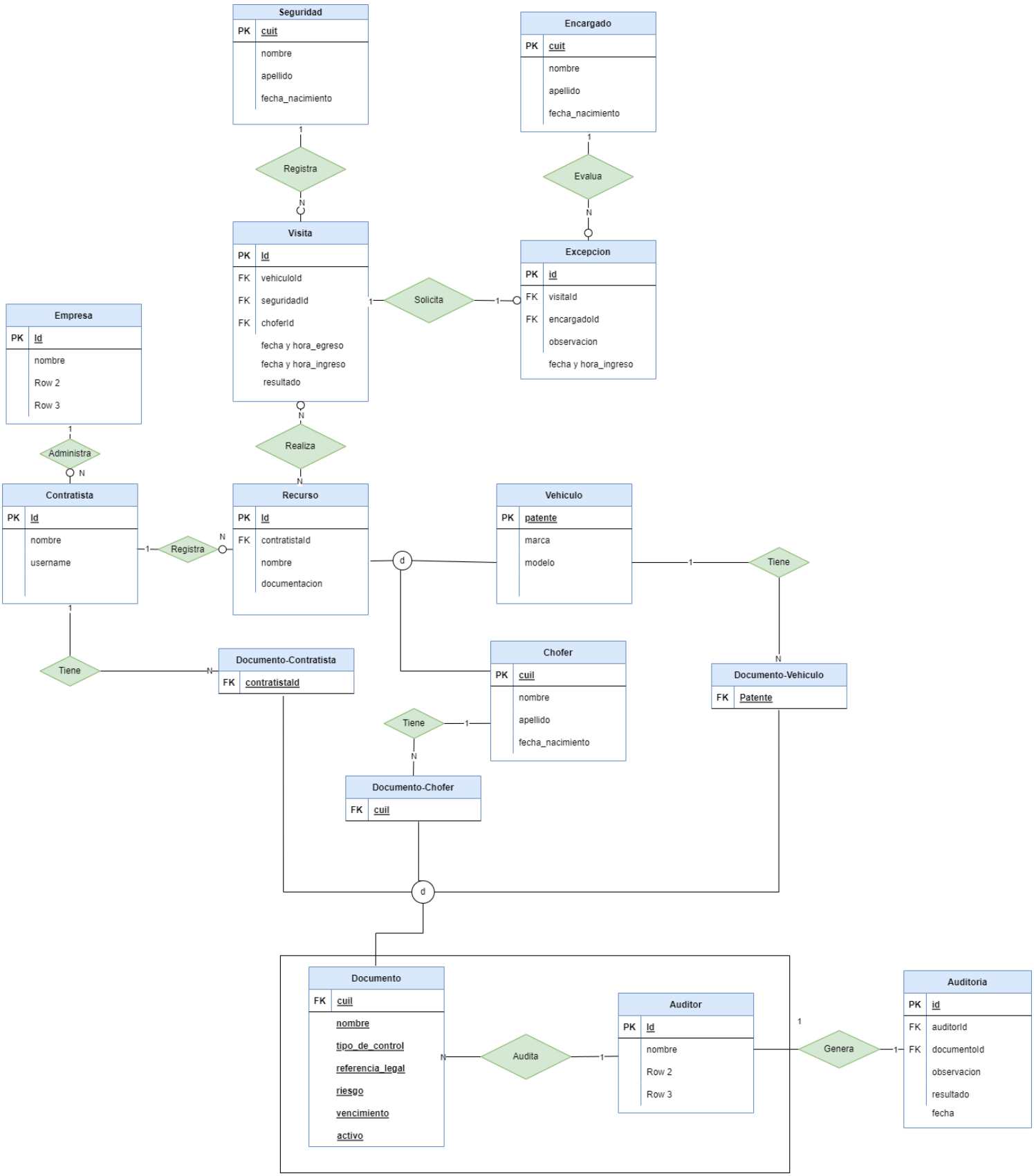
10.2 Diseño

10.2.1 Arquitectura del servidor



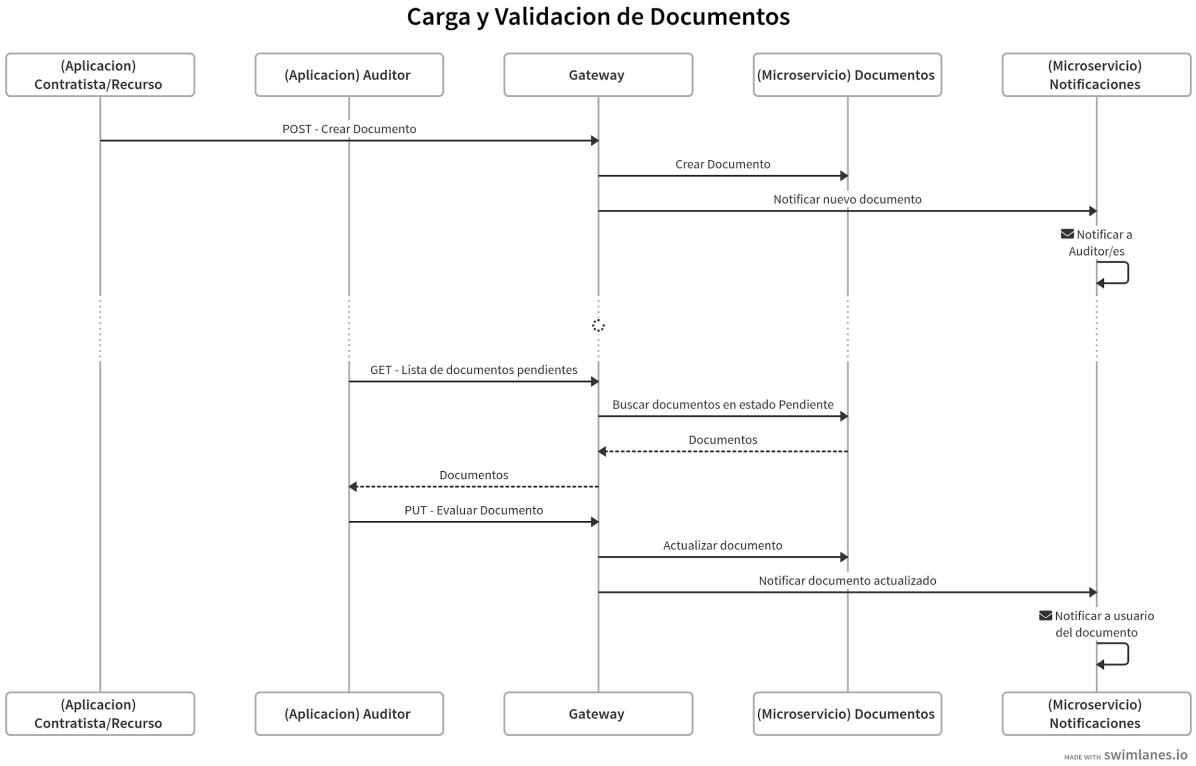
Base de datos

Diagrama de entidad-relación


Seguridad
PK cuit
nombre
apellido
fecha_nacimiento
Encargado
PK cuit
nombre
apellido
fecha_nacimiento
Registra
Evalua
Visita
PK Id
FK vehiculoId
FK seguridadId
FK choferId
fecha y hora_egreso
fecha y hora_ingreso
resultado
Solicita
Excepcion
PK id
FK visitaId
FK encargadoId
observacion
fecha y hora_ingreso
Empresa
PK Id
nombre
Row 2
Row 3
Administra
Realiza
Contratista
PK Id
nombre
username
Registra
Recurso
PK Id
FK contratistaId
nombre
documentacion
Vehiculo
PK patente
marca
modelo
Tiene
Tiene
Documento-Contratista
FK contratistaId
Chofer
PK cuil
nombre
apellido
fecha_nacimiento
Documento-Vehiculo
FK Patente
Tiene
Documento-Chofer
FK cuil
d
d
Documento
FK cuil
nombre
tipo_de_control
referencia_legal
riesgo
vencimiento
activo
Audita
Auditor
PK Id
nombre
Row 2
Row 3
Genera
Auditoria
PK id
FK auditorId
FK documentoId
observacion
resultado
fecha

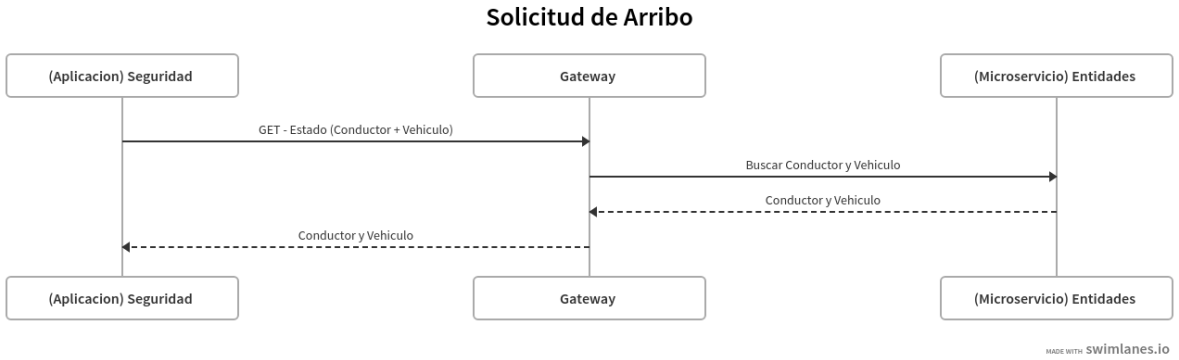
Diagramas de secuencia

Carga y Validación de Documentación



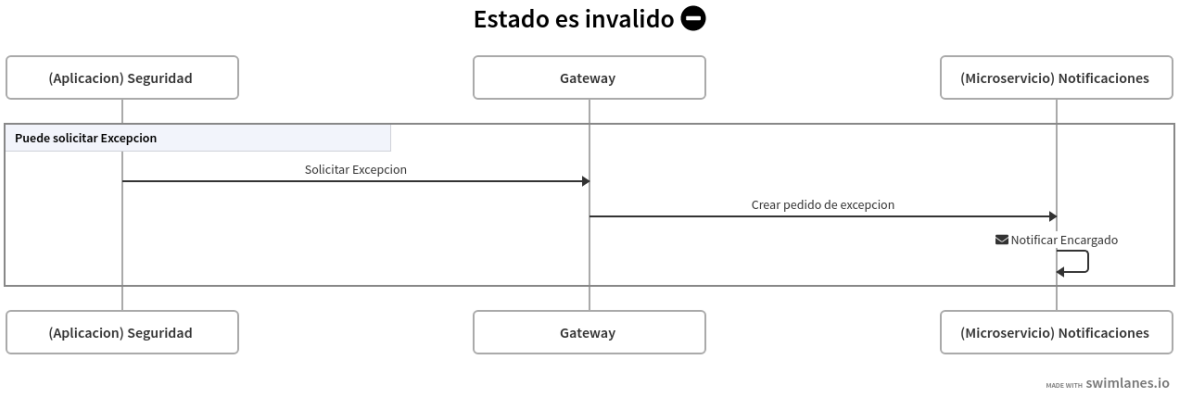
Ingreso a Planta

Solicitud de Arribo

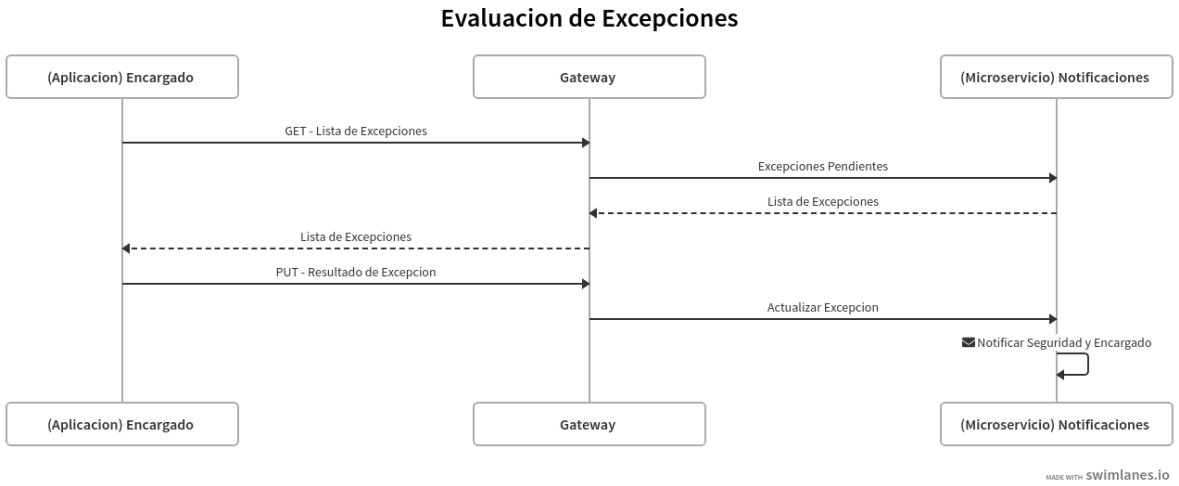


Estado Valido

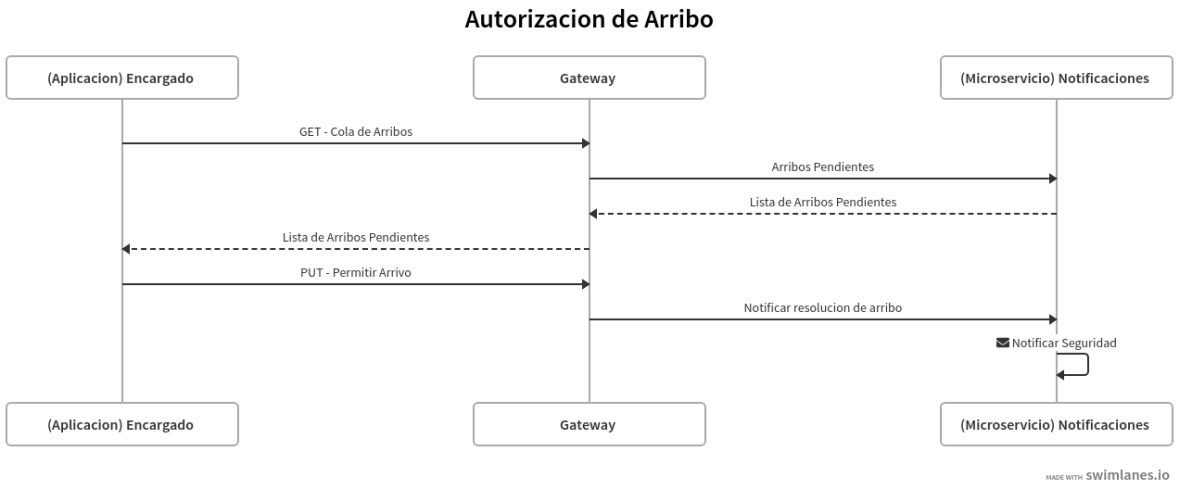
Estado Invalido



Evaluación de Excepciones



Autorización de Arribo



Creación de Visita