DISEÑO E IMPLEMENTACIÓN DE UN CONSTRUCTOR DE FORMULARIOS WEB PARA EL PROCESO DE ONBOARDING Y VALIDACIÓN DE IDENTIDAD EN TRUORA

Modalidad pasantía

Valentina Cobo Bastidas

Universidad del Valle
Facultad de ingeniería
Escuela de ingeniería de sistemas y computación
Tuluá, Valle Del Cauca
2025

DISEÑO E IMPLEMENTACIÓN DE UN CONSTRUCTOR DE FORMULARIOS WEB PARA EL PROCESO DE ONBOARDING Y VALIDACIÓN DE IDENTIDAD EN TRUORA

Modalidad pasantía

Valentina Cobo Bastidas
Código 202060174
valentina.cobo@correounivalle.edu.co

Director
Héctor Fabio Ocampo Arbeláez
hector.ocampo@correounivalle.edu.co

Co-Director
Jackson Jader Palacios Salazar, Ingeniero en Sistemas
jpalacios@truora.com

Universidad del Valle
Facultad de ingeniería
Escuela de ingeniería de sistemas y computación
Tuluá, Valle del Cauca
2025

Tabla de Contenido

Abstract

This document details the design and implementation of a web form builder at Truora, a technology company dedicated to digital identity validation and fraud prevention. The need for this tool was identified because clients had to rely on external providers for the creation of forms, which prevented Truora from centralizing the capture of valuable information for user profiling.

The project is part of Truora's strategy for 2024, seeking to consolidate its position as an integral supplier. The scope included all stages of software development, from detailed requirements gathering, architecture and database design (using AWS and DynamoDB), to the development of the web form builder with technologies such as Go (Golang) for the backend and Vue.js 3, TypeScript, Flowbite and Tailwind CSS for the frontend. A Scrumban methodology was followed, with close collaboration with the product and UI/UX team.

A Form Builder was developed that allows the creation of sections and questions with various types of input, field reordering and configurable validations. A flow was established for the storage and consultation of answers, visible on the Truora dashboard. The validation of the operation and integration was performed through unit and functional tests, both in the frontend and backend, ensuring compatibility with existing Truora systems. Finally, deployment and monitoring was performed in the AWS production environment.

Resumen

Este documento detalla el diseño e implementación de un constructor de formularios web en Truora, una empresa tecnológica dedicada a la validación de identidad digital y prevención de fraude. Se identificó la necesidad de esta herramienta debido a que los clientes debían recurrir a proveedores externos para la creación de formularios, lo que impedía a Truora centralizar la captura de información valiosa para el perfilamiento de usuarios.

El proyecto se enmarca en la estrategia de Truora para 2024, buscando consolidar su posición como proveedor integral. El alcance incluyó todas las etapas de desarrollo del software, desde el levantamiento detallado de requisitos, el diseño de la arquitectura y la base de datos (utilizando AWS y DynamoDB), hasta el desarrollo del constructor de formularios web con tecnologías como Go (Golang) para el backend y Vue.js 3, TypeScript, Flowbite y Tailwind CSS para el frontend. Se siguió una metodología Scrumban, con colaboración cercana con el equipo de producto y UI/UX.

Se desarrolló un Form Builder que permite la creación de secciones y preguntas con diversos tipos de input, reordenamiento de campos y validaciones configurables. Se estableció un flujo para el almacenamiento y consulta de respuestas, visibles en el dashboard de Truora. La validación del funcionamiento e integración se realizó a través de pruebas unitarias y funcionales, tanto en el frontend como en el backend, asegurando la compatibilidad con los sistemas existentes de Truora. Finalmente, se procedió al despliegue y monitoreo en el entorno de producción de AWS.

1. Introducción

La identidad digital y su seguridad se han consolidado como elementos esenciales en el entorno tecnológico actual, siendo de particular relevancia en América Latina, una región que enfrenta desafíos únicos en términos de seguridad digital y fraude de identidad. Truora, una empresa líder en tecnología, se ha dedicado a combatir estas amenazas mediante soluciones de vanguardia para garantizar la seguridad de las identidades y transacciones de los usuarios.

Como parte de su visión estratégica para 2024, Truora busca fortalecer su ventaja competitiva, estableciéndose como un proveedor integral de servicios de seguridad digital. Para lograr este objetivo, se ha implementado un nuevo servicio de creación de formularios web. Esta herramienta no solo aborda la necesidad de los clientes de dejar de depender de proveedores externos para la creación de formularios, sino que también optimiza y centraliza la recopilación de datos de perfilamiento, eliminando la necesidad de coordinar con múltiples terceros y agilizando los flujos de trabajo de negocio.

Este proyecto se centró en el diseño y desarrollo de dicho constructor de formularios web. Estos formularios sirven como una herramienta vital en el proceso de onboarding y validación de identidad, permitiendo a Truora recopilar de manera efectiva la información necesaria para el perfilamiento de usuarios. Además, el servicio de formularios web automatiza la captura de datos, ahorrando tiempo y recursos valiosos, y mejorando la eficiencia general del sistema.

El desarrollo abarcó desde el levantamiento detallado de requisitos, el diseño de la arquitectura y la base de datos (utilizando AWS y DynamoDB), hasta la implementación del constructor de formularios web con tecnologías como Go (Golang) para el backend y Vue.js 3, TypeScript, Flowbite y Tailwind CSS para el frontend. Se siguió una metodología Scrumban, con colaboración cercana con el equipo de producto y UI/UX. La validación del funcionamiento y la integración se realizaron a través de pruebas unitarias y funcionales en ambos stacks, asegurando la compatibilidad con los sistemas existentes de Truora, para finalmente proceder al despliegue y monitoreo en el entorno de producción de AWS.

Con esta iniciativa, Truora no solo fortalece sus propios protocolos de seguridad y prevención de fraude, sino que también establece un nuevo estándar en la industria tecnológica de América Latina, reafirmando su compromiso con la seguridad y la innovación.

2. Formulación del problema

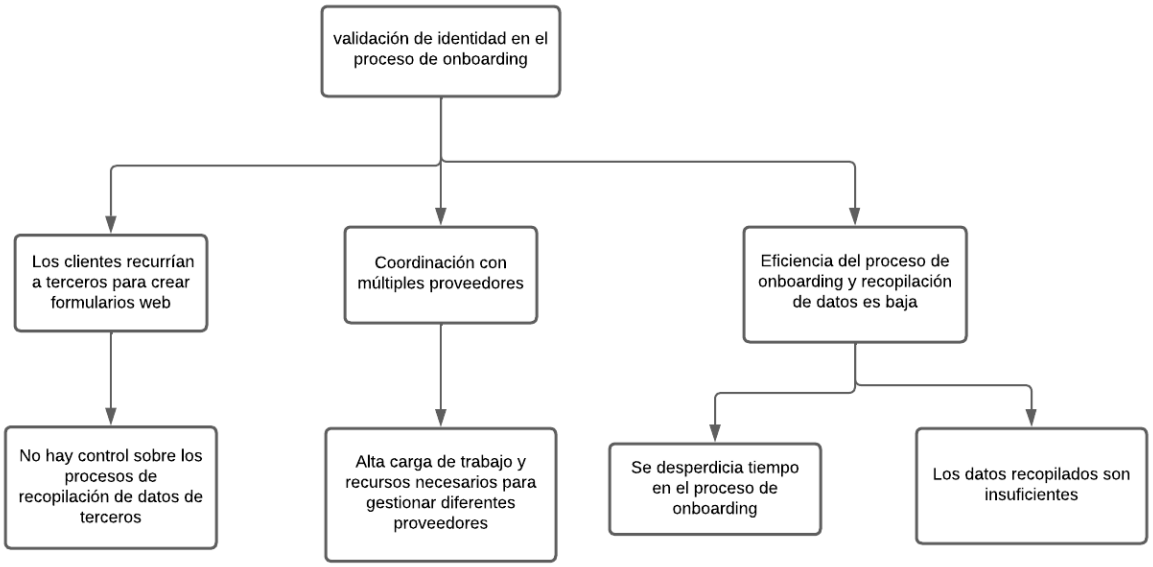

Figura 1: Árbol de problemas

2.1. Descripción del problema

En su camino para convertirse en una plataforma integral de onboarding para sus clientes, Truora se encuentra con dos desafíos importantes:

- **Carencia de un constructor de formularios Web:** En particular, los clientes inmersos en procesos de transformación digital, que suelen tener una limitada capacidad tecnológica y poca familiaridad con los procesos digitales, se ven afectados. Se ven en la necesidad de buscar proveedores externos para integrar formularios web a la plataforma de Truora. Como consecuencia, estos proveedores se convierten en el núcleo del proceso de onboarding, relegando a Truora a un rol secundario.
- **Pérdida de Información Valiosa:** La inexistencia de un constructor de formularios web impide a Truora capturar y aprovechar información vital de perfilamiento del usuario. Esta información, contenida en gran medida en las respuestas a las preguntas de perfilamiento y no solo en su identidad digital, es esencial para enriquecer los perfiles de usuario.

Estas dificultades representan significativas oportunidades de mejora en el proceso de onboarding y validación de identidad en Truora. La incorporación de un servicio de creación de formularios web solucionaría estos problemas, permitiendo una captura de datos de perfilamiento de usuario centralizada y eficiente. Esto no solo mejoraría la experiencia del cliente, sino que también reforzaría la posición de Truora como un proveedor integral

de servicios de onboarding. Además, al brindar todas las herramientas necesarias en un solo lugar, Truora se convertiría en un socio comercial difícil de reemplazar, fortaleciendo aún más su ventaja competitiva.

2.2. Definición del problema

¿Cómo el desarrollo de un constructor de formularios web puede contribuir a mejorar el proceso de onboarding y validación de identidad en Truora, permitiendo a los clientes diseñar sus propios formularios y optimizar la experiencia del usuario final?

3. Contexto de la organización

3.1. Visión

Simplificar las interacciones digitales de nuestros clientes.

3.2. Misión

Ayudar a las empresas a hacer crecer su negocio de forma segura y eficiente.

3.3. Valores Organizacionales

- **Resiliencia:** Capacidad de enfrentar desafíos, superar obstáculos y mantener la motivación en el entorno de trabajo.
- **Liderar con el ejemplo:** Demostrar los valores, habilidades y comportamientos que se espera ver en los demás, sirviendo como modelo a seguir y generando inspiración y confianza en su equipo.
- **Responsabilidad:** Asumir la responsabilidad de una tarea, proyecto o situación y actuar de manera proactiva para garantizar su éxito.
- **Confianza:** Creer en la integridad, capacidad y honestidad de los demás, así como en su compromiso para cumplir con las responsabilidades y alcanzar los objetivos.
- **Hablar directamente:** Expresar opiniones, ideas o feedback de forma clara y honesta, sin ocultar información o suavizar el mensaje.

4. Descripción de la empresa

4.1. Información general

Truora es una startup latinoamericana fundada en 2018 que se especializa en soluciones de verificación de antecedentes y seguridad para empresas y plataformas digitales. Ofrece servicios como verificación de identidad, antecedentes penales, referencias laborales y educativas, entre otros. Truora utiliza tecnología avanzada, como inteligencia artificial y machine learning, para agilizar y mejorar el proceso de verificación, ayudando a las empresas a tomar decisiones informadas al evaluar a candidatos o usuarios. Su objetivo es proporcionar confianza y seguridad en las transacciones y relaciones comerciales en línea.

4.2. Razón social de la empresa

Truora S.A.S. - NIT 9011899795

4.3. Tipo de empresa

Empresa de tecnología.

4.4. Equipos dentro de la empresa

La empresa está conformada por los siguientes equipos de trabajo:

- Siete equipos de ingeniería enfocados en el desarrollo front-end y back-end tanto para aplicaciones web como móviles.
- Equipo de ventas.
- Equipo de producto.
- Equipo de diseñó UI/UX.
- Equipo de growth hacking.

4.5. Número de empleados

El equipo de Truora consta de 104 empleados.

4.6. Mapa de procesos

En la Figura 1 se puede observar el mapa de procesos de la empresa.

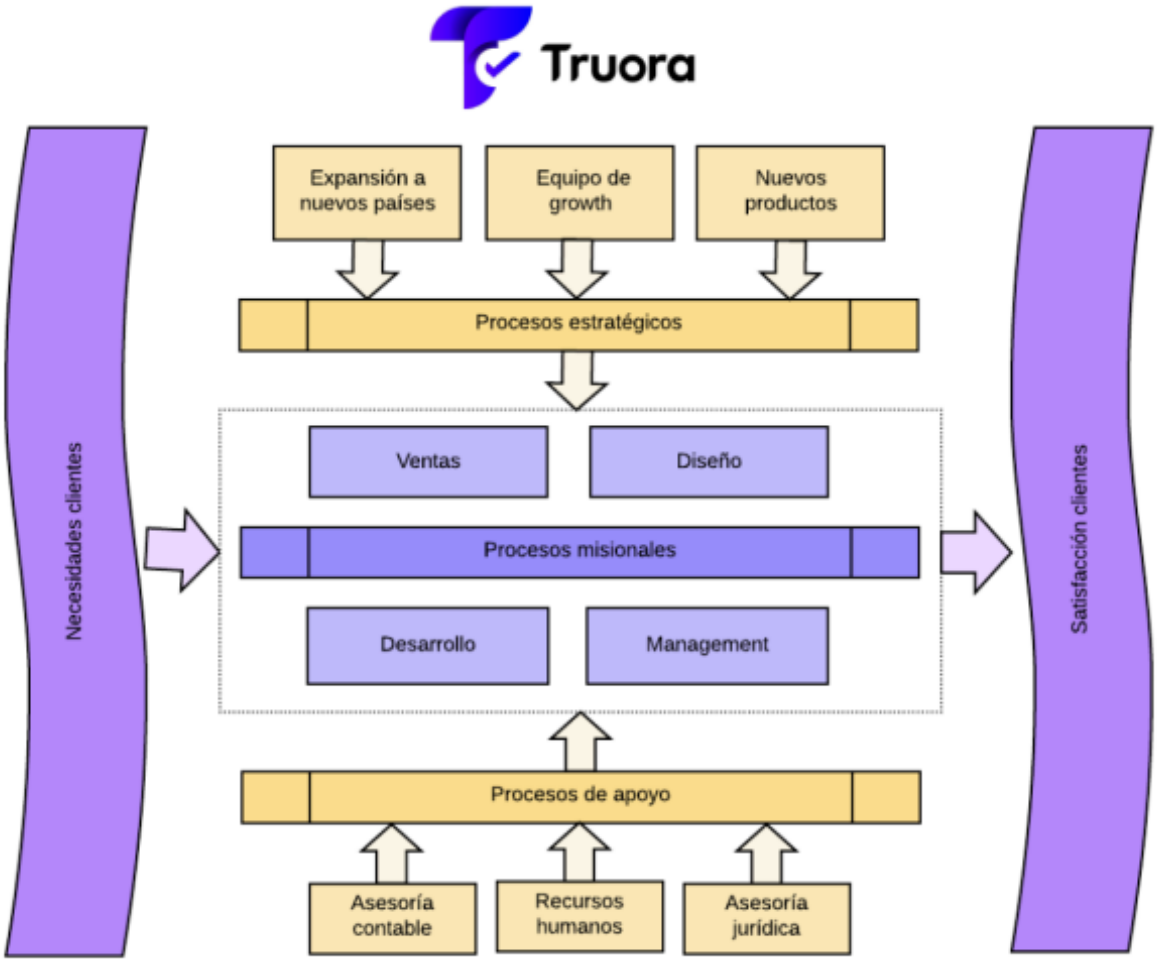

Figura 2: Mapa de procesos empresa Truora.

4.7. Procesos involucrados en el proyecto que el pasante participará

La ejecución de este proyecto necesita una cercana relación con los pasos de crecimiento de la empresa. Esto va desde la etapa de planear e implementar, hasta probar y poner en marcha el nuevo servicio. Además, el estudiante en prácticas tendrá la oportunidad de involucrarse directamente en varios procesos de diseño de interfaces gráficas que serán una parte esencial del nuevo producto.

5. Descripción de las funciones a desarrollar en la organización

El proyecto se trabajará en un equipo de cuatro personas, siendo estas las tareas del estudiante:

Tareas del backend:

- Diseño e implementación de la arquitectura del servicio en la nube utilizando AWS y Terraform. Esto incluye la creación y gestión de instancias EC2, configuración de grupos de seguridad, políticas IAM, y otros servicios necesarios.
- Desarrollo de servicios RESTful en Golang para el manejo de formularios. Esto incluirá servicios para crear, listar, editar y borrar formularios.
- Implementación de la lógica de negocio en Golang para el procesamiento de los formularios.
- Creación de pruebas unitarias y de integración para el servicio de formularios.
- Creación y gestión de tablas en DynamoDB para almacenar los formularios y las respuestas.
- Configuración y administración de Route53 para manejar el DNS del servicio.
- Creación y administración de contenedores Docker para el despliegue del servicio.

Tareas del frontend:

- Diseño e implementación de componentes Vue 3 para la interacción con los formularios. Esto incluirá interfaces para crear, editar, listar y rellenar formularios.
- Implementación de la lógica de negocio en TypeScript para procesar las interacciones del usuario con los formularios.
- Utilización de Tailwind CSS y Flowbite para el diseño y estilización de las interfaces de usuario.
- Creación de pruebas unitarias y de integración para los componentes Vue y las funciones TypeScript.
- Integración de la interfaz de usuario con los servicios backend mediante el uso de llamadas a la API.
- Implementación de prácticas de seguridad y optimización del rendimiento de la aplicación.

6. Competencias a desarrollar del estudiante en la organización

6.1. Competencias personales

- **Colaboración en equipo:** En este proyecto, los participantes podrán mejorar sus habilidades de trabajo en grupo al colaborar con otros miembros. Esto les permitirá fortalecer sus capacidades comunicativas, resolución de problemas, adaptabilidad y eficiencia en un ambiente de trabajo conjunto.

- **Competencias técnicas:** La participación en el desarrollo e implementación del software brindará a los estudiantes la oportunidad de potenciar sus habilidades técnicas en áreas como la programación, diseño de algoritmos o desarrollo web. De esta forma, podrán obtener experiencia práctica y reforzar su formación en el área tecnológica.

- **Capacidades de investigación:** Durante la realización del proyecto, los estudiantes tendrán la necesidad de indagar y examinar diversas alternativas, tecnologías o metodologías para afrontar los retos que se presenten. Este proceso les permitirá desarrollar habilidades de investigación, análisis de información y toma de decisiones basadas en datos.

- **Manejo del tiempo y organización:** La participación en el proyecto les exigirá a los estudiantes manejar eficientemente su tiempo y organizar sus tareas respetando los plazos establecidos. Esto les ayudará a adquirir habilidades en gestión del tiempo, establecimiento de prioridades y logro de metas.

- **Razonamiento crítico:** A lo largo del proyecto, los estudiantes tendrán la oportunidad de fortalecer su pensamiento crítico al enfrentar desafíos y tomar decisiones. Esto les permitirá evaluar distintas situaciones, identificar problemas, proponer soluciones innovadoras y adoptar decisiones informadas basándose en la información disponible.

6.2. Competencias profesionales

- **Enfoque en el cliente y calidad:** El proyecto busca realzar la experiencia del usuario y la calidad de las imágenes obtenidas. Este enfoque permitirá desarrollar habilidades centradas en el cliente, comprendiendo sus necesidades y brindando soluciones que cumplen con estrictos estándares de calidad.

- **Desarrollo de software:** Participando en este proyecto, los estudiantes podrán poner en práctica sus conocimientos en la creación de software, y ganarán experiencia práctica en la implementación de soluciones tecnológicas. Esto mejorará sus habilidades de programación, diseño de algoritmos y resolución de problemas.

- **Metodologías ágiles:** En el proyecto, los estudiantes tendrán la invaluable oportunidad de adentrarse en las metodologías ágiles, adquiriendo destrezas clave en este enfoque flexible y colaborativo para la gestión de proyectos. Durante esta experiencia, podrán aprender y aplicar varios conceptos y prácticas de metodologías ágiles, como Scrum y Kanban, para optimizar el proceso de desarrollo de software.

7. Alcance del Proyecto

El propósito de este proyecto es esencial en la visión estratégica de Truora para el año 2024. No se trata simplemente de la creación de un constructor de formularios web, sino de un pilar fundamental en el objetivo de fortalecer la posición de Truora como un proveedor integral de servicios de onboarding. Este proyecto es un elemento crucial en la iniciativa para expandir nuestro portafolio de productos y, de este modo, reforzar nuestra presencia irremplazable en el mercado.

El alcance del proyecto abarca todas las etapas de desarrollo del software, desde el diseño de la arquitectura y las bases de datos necesarias hasta el despliegue y mantenimiento de la primera versión del constructor de formularios web. Este proyecto, que se desarrollará a lo largo de 8 meses de trabajo a tiempo parcial, tiene como meta mejorar el proceso de onboarding y proporcionar un formulario web intuitivo y fácil de usar para nuestros clientes.

Un aspecto fundamental de este proyecto es su contribución al perfilamiento de los usuarios. A través de este constructor, obtendremos información valiosa que otros validadores no nos pueden proporcionar. Esta información será crucial para mejorar la precisión y eficacia de nuestros servicios de identidad digital.

Además, nuestro constructor de formularios web se integrará eficientemente con los sistemas existentes de Truora, asegurando una recopilación de datos de usuario más centrada y efectiva. Este enfoque integral eliminará la necesidad de que nuestros clientes busquen proveedores externos para la creación de formularios web, aumentando así el valor de los servicios de Truora.

De cara al futuro, este proyecto no solo ayudará a Truora a consolidarse como un proveedor esencial de servicios de onboarding, sino que también permitirá la integración completa de todos nuestros servicios en nuestra plataforma. Esto hará que el servicio de identidad digital de Truora sea aún más potente, mejorando la experiencia del cliente y la eficiencia operativa, y acercándonos a nuestra meta para el año 2024.

7.1. Supuestos

- Truora proporcionará todos los recursos necesarios, humanos y técnicos, para el desarrollo del proyecto.
- Todo el equipo de proyecto estará disponible y dedicará el tiempo necesario para la realización del proyecto durante los 8 meses estipulados.
- El equipo de proyecto posee las habilidades y competencias necesarias para el desarrollo de todas las etapas del software.
- El constructor de formularios web podrá integrarse sin problemas con los sistemas existentes de Truora.

- Los clientes de Truora aceptarán y utilizarán este nuevo constructor de formularios web para el proceso de onboarding.
- Todo el desarrollo del proyecto cumplirá con las legislaciones y normativas en materia de protección de datos.

7.2. restricciones

- Debido a la naturaleza sensible de la información recopilada a través de los formularios, se requiere un manejo cuidadoso y seguro de estos datos. Esto implica el uso de protocolos de seguridad adecuados y el cumplimiento de las normativas de protección de datos.
- El nuevo servicio debe ser diseñado e implementado de tal manera que sea compatible con la plataforma existente de Truora. Esto es vital para garantizar una integración sin problemas y un funcionamiento eficaz.
- Aunque el servicio es independiente, debe poder utilizarse dentro de los flujos de validación de identidad existentes para mejorar el perfilamiento de usuarios. Esto significa que el servicio debe diseñarse pensando en su integración en estos flujos.
- Dado que el proyecto se desarrollará a tiempo parcial durante 8 meses, el tiempo puede ser una restricción. Cumplir con los plazos establecidos sin comprometer la calidad del producto final puede ser un desafío.
- Los recursos disponibles, tanto financieros como humanos, pueden restringir el alcance o la velocidad del proyecto.

7.3. Objetivos

7.3.1. Objetivo general

Desarrollar un constructorde formularios web para mejorar el proceso de onboarding y validación de identidad en Truora, con el fin de fortalecer su posición como un proveedor integral de servicios de onboarding, y optimizar la experiencia del cliente.

7.3.2. Objetivos específicos

- Realizar un levantamiento detallado de los requerimientos necesarios para la creación del constructor de formularios web, considerando tanto las necesidades del cliente como las de la empresa.
- Diseñar la estructura del software que cumpla con los requerimientos establecidos y se acople a la arquitectura del software de la empresa.
- Desarrollar el constructor de formularios web y asegurar su integración efectiva con los sistemas existentes de Truora.

- Validar el funcionamiento del constructor de formularios web y su integración con los sistemas de Truora.
- Desarrollar y ejecutar un conjunto de pruebas completas en el software para asegurar que todos los requerimientos establecidos inicialmente se han implementado correctamente.
- Desplegar y monitorear el constructor de formularios web en el entorno de producción asegurando el correcto funcionamiento con todos los servicios.

7.3.3. Resultados esperados

Tabla 1: Tabla de objetivos específicos y su resultado

Objetivo especifico	Resultado
Realizar un levantamiento detallado de los requerimientos necesarios para la creación del constructor de formularios web, considerando tanto las necesidades del cliente como las de la empresa.	▪ Documento de requerimientos. ▪ Historias de usuario.
Diseñar la estructura del software que cumpla con los requerimientos establecidos y se acople a la arquitectura del software de la empresa.	▪ Diagramas de diseño. ▪ Diagramas de flujo. ▪ Definición de tecnologias a utilizar.
Desarrollar el constructor de formularios web y asegurar su integración efectiva con los sistemas existentes de Truora.	▪ Código fuente. ▪ Guía de uso para desarrolladores. ▪ Infraestructura en AWS
Validar el funcionamiento del formulario web y su integración con los sistemas de Truora.	▪ Informe de resultados de pruebas.
Desarrollar y ejecutar un conjunto de pruebas completas en el software para asegurar que todos los requerimientos establecidos inicialmente se han implementado correctamente.	▪ Código fuente. ▪ Documento de resultados de pruebas exhaustivas.
Desplegar y monitorear el constructor de formularios web en el entorno de produccion asegurando el correcto funcionamiento con todos los servicios.	▪ Código fuente. ▪ Alarmas en Kibana ▪ Pruebas calendarizadas

8. Desarrollo del proyecto

8.1. Levantamiento Detallado de Requerimientos para la Creación del constructor de formularios Web

Para el levantamiento de requerimientos en la creación de un constructor de formularios web dentro de Truora se llevó a cabo con el objetivo de desarrollar una solución que no solo responda a las necesidades de la empresa, sino también a las de sus clientes. Este proceso fue fundamental para asegurar que el producto final cumpliera con las expectativas tanto internas como externas y se integrara de manera eficiente con la arquitectura existente.

8.1.1. Identificación de la Necesidad

Truora ha identificado una oportunidad para expandir su oferta de servicios mediante el desarrollo de un producto que permite a sus clientes crear formularios dinámicos con facilidad. Este nuevo producto se integraria directamente en sus flujos de verificación de identidad, lo que facilitara la captura de información clave durante el proceso de onboarding. A diferencia de los formularios convencionales, las respuestas recopiladas se pueden utilizar como "variables" en otros validadores dentro del proceso de verificación, lo que añade un valor significativo al ecosistema de verificación digital de Truora.

Además, este producto proporciona la capacidad de configurar automatizaciones a través del servicio de Business Rules Engine (BRE) de Truora. Por ejemplo, cuando se completa un formulario, se pueden desencadenar acciones automáticas, como el envío de un mensaje de WhatsApp al número de teléfono proporcionado. Esta funcionalidad no solo mejora la eficiencia operativa, sino que también destaca la flexibilidad y el control que Truora ofrece a sus clientes, permitiendo el manejo de información crítica sin depender de proveedores externos.

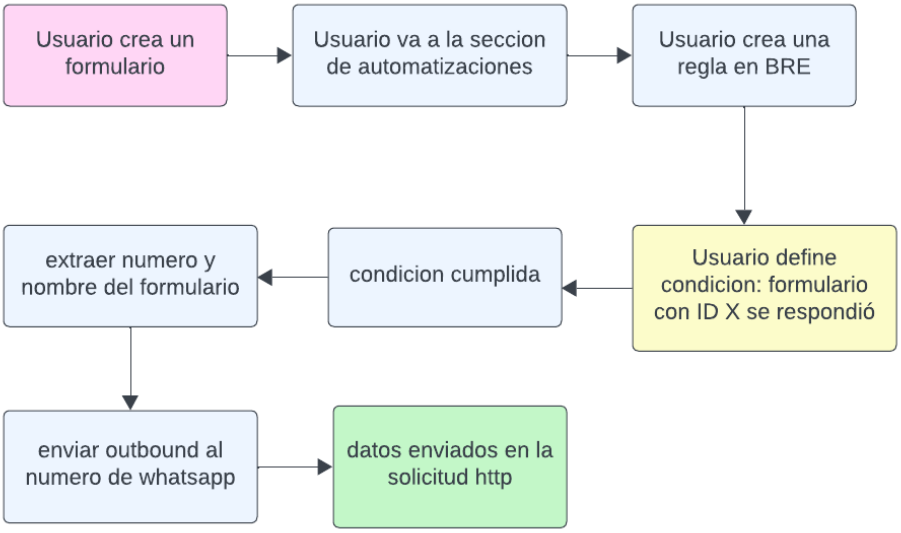

Figura 3: Diagrama de flujo (integracion con el servicio de BRE).

8.1.2. Reunión con el Cliente y Análisis de Soluciones Externas

El proceso de levantamiento de requerimientos se inició con una reunión clave con uno de los clientes. Durante este encuentro, se presentó un formulario externo que había sido previamente integrado con los servicios de la compañía. Esta presentación funcionó como punto de referencia para identificar las funcionalidades necesarias que debía incluir la solución interna. A partir de esta reunión, se delinearon las principales características que debía contemplar el constructor de formularios web, ajustadas a las necesidades específicas del cliente y con el objetivo de garantizar una integración efectiva con los servicios existentes.

8.1.3. Requerimientos del Sistema

ID	Nombre del Requisito	Descripción
RF-01	Integración con Identidad Digital	El sistema debe permitir que los formularios web se integren como parte del flujo de validación de identidad digital existente, actuando como un componente dentro del conjunto de validadores y verificadores.
RF-02	Uso Independiente de Formularios	El sistema debe permitir que los formularios puedan ser utilizados sin necesidad de estar integrados a un flujo de identidad, permitiendo la captura de datos en forma autónoma.
RF-03	Compatibilidad con BRE	El sistema debe permitir que los formularios generen eventos que puedan ser leídos por el servicio de Business Rule Engine (BRE) para ejecutar reglas definidas por el cliente.
RF-04	Centralización de Datos	El sistema debe almacenar toda la información capturada por los formularios en una base de datos centralizada y accesible desde los sistemas internos de Truora.
RF-05	Descarga de Respuestas	El sistema debe permitir que los clientes accedan y descarguen las respuestas registradas por los usuarios a través de los formularios.
RF-06	Integración con Reportes	Las respuestas de los formularios deben estar disponibles en el historial de resultados de procesos WEB del producto de Identidad Digital.
RF-07	Reglas BRE desde Formularios	El sistema debe permitir al cliente definir reglas en BRE basadas en las respuestas a los formularios, para automatizar acciones.

Tabla 2: Requerimientos Funcionales

ID	Nombre del Requisito	Descripción
RNF-01	Compatibilidad Arquitectónica	El sistema debe ser compatible con la arquitectura de microservicios existente de Truora.
RNF-02	Escalabilidad	El sistema debe permitir el procesamiento concurrente de múltiples formularios sin afectar el rendimiento.
RNF-03	Seguridad	El sistema debe garantizar el cifrado de los datos sensibles capturados por los formularios.
RNF-04	Trazabilidad	El sistema debe registrar un log con fecha/hora de cada interacción con los formularios (envío, edición, descarga).

Tabla 3: Requerimientos No Funcionales

ID	Restricción	Descripción
RC-01	Uso de servicios existentes	Los formularios deben integrarse con servicios existentes como Identidad Digital y BRE, sin modificar su comportamiento base.
RC-02	Lenguaje y tecnologías	El desarrollo debe realizarse en Node.js para las funciones lambda, siguiendo los lineamientos internos de infraestructura.
RC-03	Estándares de Interfaz	Deben seguirse los estándares definidos por Truora para componentes web embebidos.

Tabla 4: Restricciones del Sistema

8.1.4. Desarrollo de historias de usuario

Para estructurar de manera clara y concisa las necesidades identificadas, se desarrollaron una serie de user stories que capturan las funcionalidades clave desde la perspectiva del usuario final:

1. **Como cliente, quiero poder crear un formulario personalizado para capturar información personal de mis usuarios que no puede ser recolectada mediante los validadores existentes, de modo que pueda adaptar el proceso de recolección de datos a mis necesidades específicas.**
    - Creación de un workspace para reutilizar componentes.
    - Desarrollo de lambdas para las operaciones CRUD de los formularios.
    - Desarrollo del *form builder*.
2. **Como cliente, quiero usar los formularios web fuera de los flujos de validación de identidad.**

- Implementación de una vista para rellenar el formulario como producto independiente.

3. **Como cliente, quiero usar los formularios web dentro de los flujos de validación de identidad.**
    - Desarrollo de tabla de Formularios en el producto de identidad digital.
    - Integración de formularios con procesos WEB.
    - Habilitar uso de respuestas como variables en otros validadores.
4. **Como cliente, quiero acceder a la información registrada por mis usuarios en el formulario.**
    - Integración con el sistema de reportes para descargar las respuestas de un formulario.
    - Desarrollo de lambdas para las operaciones CRUD de las respuestas del formulario.
    - Disponibilidad de la información de las respuestas en el historial de resultado de los procesos WEB.
5. **Como cliente, quiero crear reglas a partir de las respuestas de los formularios para ejecutar acciones en BRE.**
    - Integración con el servicio de BRE.
    - Desarrollo de lambdas para enviar eventos al servicio de BRE.

8.2. Diseñar la estructura del software que cumpla con los requerimientos establecidos y se acople a la arquitectura del software de la empresa

8.2.1. Base de datos

Se ha elegido Amazon DynamoDB como la base de datos principal debido a los beneficios clave que ofrece y que se ajustan a las necesidades del sistema [7]. DynamoDB es una base de datos NoSQL de alto rendimiento, diseñada para manejar grandes volúmenes de datos de manera eficiente y con baja latencia. Su diseño simplificado y orientado a tablas permite realizar operaciones CRUD (crear, leer, actualizar y eliminar) de forma ágil y sin la necesidad de consultas complejas, lo cual se adapta adecuadamente a los requerimientos de los formularios web de la plataforma.

Además de la eficiencia en las operaciones básicas, DynamoDB incluye la funcionalidad de streams, la cual permite capturar cambios en los datos en tiempo real. Esta característica se utiliza para disparar eventos y notificar a otros servicios en la plataforma cuando se realizan modificaciones en las respuestas o configuraciones de los formularios. Este enfoque facilita la comunicación entre servicios sin requerir integraciones complejas, y garantiza que cada cambio en la base de datos desencadene automáticamente acciones en otros componentes.

8.2.2. Arquitectura

La arquitectura de la plataforma de formularios web se fundamenta en un enfoque de servicios distribuidos, también conocido como arquitectura multiservicios o de microservicios. Esta se ha implementado utilizando el lenguaje de programación Go, debido a sus características técnicas que lo hacen adecuado para este tipo de sistemas, como su eficiencia en el manejo de concurrencia, su buen desempeño en entornos de red y la capacidad de generar binarios livianos [1].

La adopción de una arquitectura multiservicios proporciona beneficios clave en comparación con un enfoque monolítico. En lugar de una única aplicación centralizada, la solución está dividida en múltiples servicios autónomos, cada uno encargado de una función específica. Este diseño permite que los servicios puedan desarrollarse, desplegarse y escalarse de manera independiente, lo que facilita la implementación de mejoras, actualizaciones y correcciones sin afectar el resto de la plataforma. Asimismo, se incrementa la resiliencia del sistema, ya que el fallo de un componente no compromete el funcionamiento completo. Esta estructura también favorece el trabajo en equipos distribuidos, permitiendo que cada grupo se concentre en servicios concretos de manera ágil y autónoma.

Para alojar y gestionar estos servicios, se ha utilizado Amazon Web Services (AWS) como plataforma de infraestructura en la nube [2]. AWS proporciona una amplia gama de servicios que han sido integrados en la arquitectura, entre los cuales se destacan:

- **API Gateway:** Servicio utilizado para gestionar y exponer los diferentes puntos de entrada de la API de forma centralizada, facilitando la administración de peticiones, autenticación y la escalabilidad de la comunicación entre clientes y servicios [8].
- **AWS Lambda:** Se utilizan funciones Lambda para ejecutar procesos en respuesta a eventos, como modificaciones en la base de datos, sin necesidad de gestionar servidores, lo que permite escalar automáticamente y reducir costos operativos [9].
- **DynamoDB:** Base de datos NoSQL de alto rendimiento, seleccionada por su baja latencia y su funcionalidad de *streams*, que permite activar eventos en otros servicios cuando ocurren cambios en los datos [7].
- **Contenedores (ECS o EKS):** Servicios empleados para ejecutar aquellos microservicios que requieren mayor control sobre el entorno de ejecución, permitiendo escalar aplicaciones que demandan alta capacidad de procesamiento [10].
- **S3 Buckets:** Servicio de almacenamiento utilizado para alojar archivos, como formularios generados, documentos y archivos multimedia. Amazon S3 garantiza durabilidad, seguridad y escalabilidad para el almacenamiento masivo [11].
- **SQS (Simple Queue Service):** Sistema de colas utilizado para facilitar la comunicación asincrónica entre servicios. Mediante SQS, es posible desacoplar componentes, permitiendo que los mensajes sean procesados independientemente del tiempo en que fueron generados, lo cual mejora la tolerancia a fallos [12].
- **CloudFront:** Servicio de distribución de contenido que mejora la velocidad de entrega y proporciona funcionalidades de monitoreo y seguridad, como la gestión de *logs* y auditorías [13].
- **CloudWatch:** Herramienta de monitoreo que permite supervisar el rendimiento de los servicios y recolectar métricas en tiempo real, facilitando el análisis de fallos y el mantenimiento proactivo de la plataforma [14].

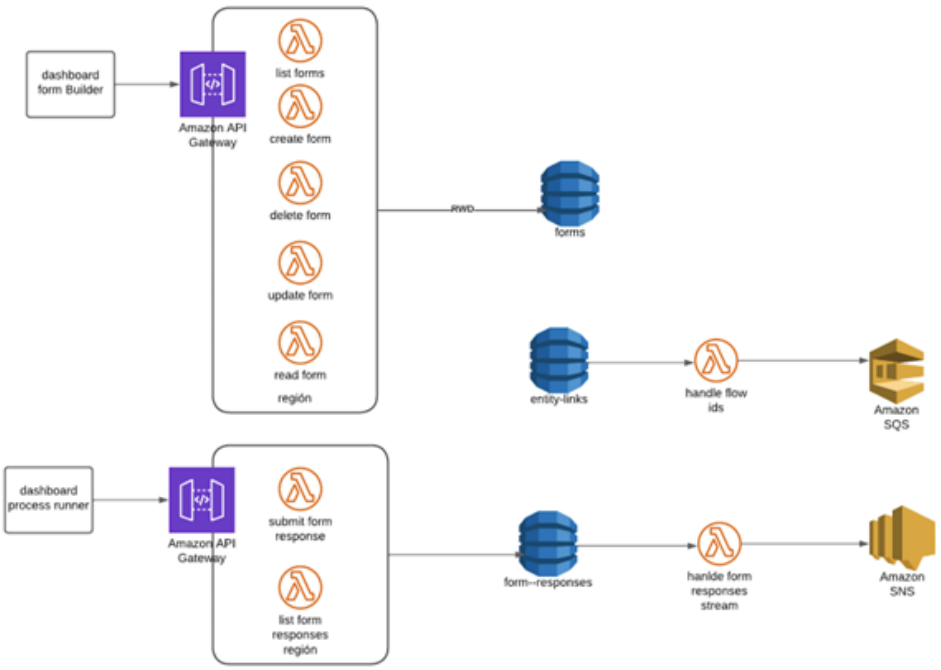

Figura 4: Arquitectura de formularios.

8.2.3. Patrón de diseño

En el proyecto *Formularios*, cada módulo adopta una combinación estratégica de arquitectura de capas y arquitectura limpia, lo que permite mantener una separación clara de responsabilidades y una organización eficiente del código. Esta combinación se materializa en los siguientes componentes clave:

- **Models:** Los modelos son fundamentales para definir las entidades del dominio y sus relaciones. Sirven como la representación lógica de los datos del sistema, asegurando una comprensión y manejo coherente de las estructuras de información. Estos modelos son agnósticos de la lógica de negocio, lo que facilita su reutilización y adaptación.

- **Storage:** Esta capa se ocupa de la persistencia de datos. Se encarga de las operaciones de almacenamiento y recuperación, interactuando con bases de datos u otros sistemas de persistencia. La arquitectura limpia asegura que esta capa sea independiente de las capas superiores, lo que significa que las decisiones de almacenamiento no afectan la lógica de negocio ni la definición de los modelos.

- **Lambda:** Aquí se concentra la lógica de negocio, actuando como intermediario entre los *Models* y *Storage*. Esta capa se encarga de implementar las reglas y procesos que definen el comportamiento del sistema. Al utilizar una arquitectura de capas, se garantiza que los cambios en la lógica de negocio no repercutan en los modelos o en la forma en que se almacenan los datos.

8.3. Desarrollo del constructor de formularios web e integración con los sistemas existentes de Truora

8.3.1. Inicio del desarrollo

El desarrollo del constructor de formularios web comenzó con una fase de definición conjunta con el equipo de producto. Durante esta etapa, se elaboró un documento de requerimientos de producto (*Product Requirements Document*, PRD), el cual contenía tanto requerimientos funcionales como un análisis de negocio asociado. A partir de dicho documento, se generó una propuesta técnica donde se tradujeron los requerimientos del producto en tareas técnicas concretas, definiendo además el alcance inicial del desarrollo.

Las funcionalidades clave a implementar se priorizaron con base en las siguientes historias de usuario:

- *Como cliente, puedo crear un formulario para capturar información de mis usuarios.*
- *Como cliente, puedo utilizar los formularios web dentro de los flujos de validación de identidad.*
- *Como cliente, puedo acceder a la información registrada por mis usuarios en el formulario.*
- *Como negocio, puedo agilizar procesos de identidad y obtener mejor entendimiento sobre mis clientes.*

En esta etapa se establecieron límites claros para el alcance del proyecto. El desarrollo se centró exclusivamente en la versión web del formulario, descartando otros entornos como WhatsApp. Asimismo, se decidió excluir la lógica condicional de esta primera versión, reservando dicha funcionalidad para futuras iteraciones.

Desde el punto de vista técnico, se adoptaron las tecnologías ya establecidas en la arquitectura existente. El backend fue desarrollado en **Go (Golang)** [1], mientras que el frontend se construyó con **Vue.js 3** [3], haciendo uso adicional de **TypeScript** [6], **Flowbite** [4] y **Tailwind CSS** [5] para el diseño visual y la reutilización de componentes. Para la gestión del estado global en el frontend se utilizó **Pinia** [15].

El constructor de formularios fue integrado dentro de un panel ya existente denominado **Console**, al cual se incorporaron nuevas rutas, vistas, *stores* y componentes específicos para esta funcionalidad.

El desarrollo fue ejecutado en colaboración con el equipo **Regulus**, conformado por tres ingenieros de software y un Product Manager. El equipo adoptó una metodología ágil del tipo **Scrumban**, trabajando en *sprints* semanales con cierres los días martes. Durante cada ciclo se llevaron a cabo reuniones diarias (*dailies*), retrospectivas al finalizar cada

sprint, y sesiones de planificación (*plannings*) para estimar y priorizar tareas. La comunicación continua se mantuvo mediante un grupo de Telegram, lo que facilitó la coordinación operativa de manera ágil.

8.3.2. Desarrollo del Form Builder

El **Form Builder** constituye el núcleo del producto desarrollado. Su propósito es permitir a los usuarios crear formularios web de manera sencilla, visual, flexible e intuitiva. Entre sus funcionalidades principales se encuentran:

- Creación de múltiples secciones dentro de un formulario.
- Adición de preguntas con diversos tipos de *input*.
- Reordenamiento de preguntas o movimiento entre secciones mediante *drag and drop*.
- Clonado o eliminación de preguntas existentes.
- Vista previa interactiva del formulario final.
- Edición del nombre del formulario.
- Configuración del formulario para ser usado en flujos de validación de identidad de Truora.

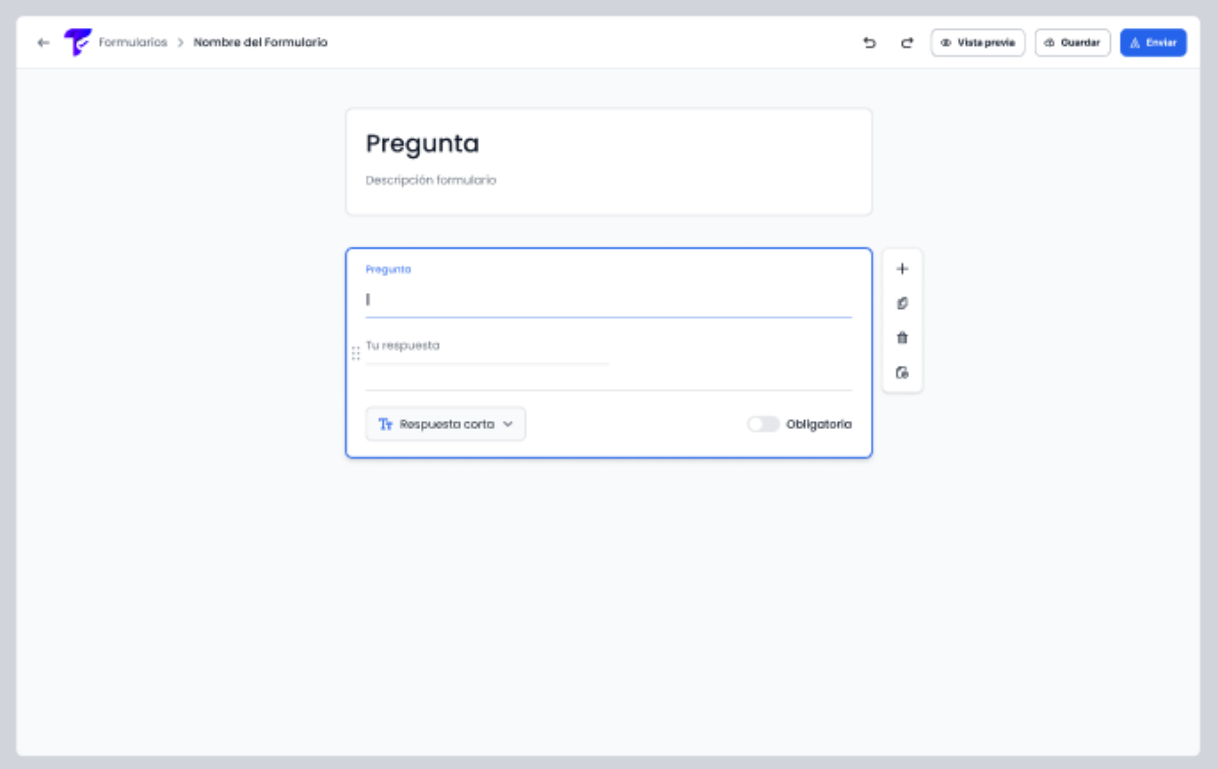

Figura 5: Vista principal del FormBuilder.

Figura 6: Acciones dentro del FormBuilder.

Estas funciones están organizadas en una interfaz accesible desde la barra de navegación, que incluye acciones como guardar, previsualizar, cambiar nombre y enviar el formulario.

El constructor permite a los usuarios agregar una variedad de campos, entre ellos:

- Texto corto (*short_answer*)
- Texto largo (*long_text*)
- Correo electrónico (*email*)
- Número (*numeric*)
- Fecha (*date*)
- Teléfono (*phone*)
- Lista desplegable (*list*)
- Selección múltiple (*checkbox*)
- Selección única (*radio*)

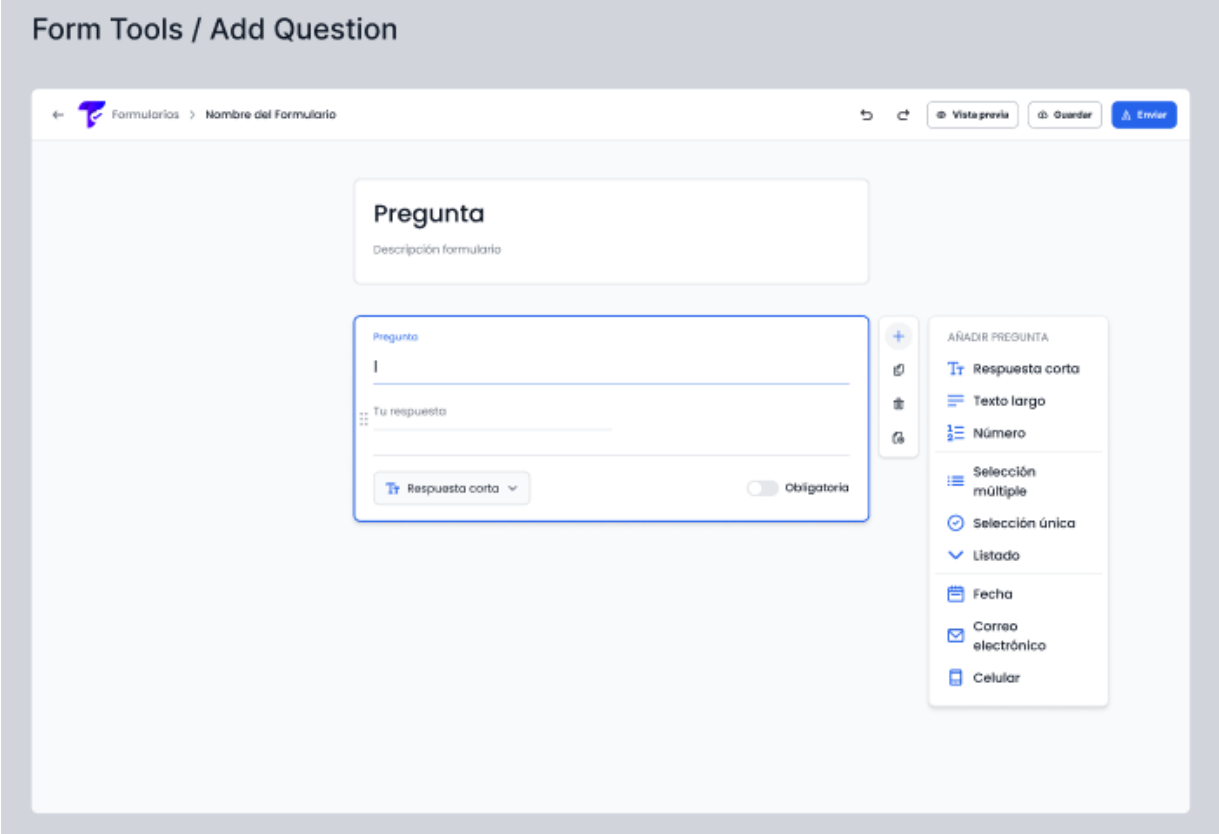

Figura 7: Acciones dentro del FormBuilder.

Cada campo incluye validaciones configurables para asegurar la calidad de los datos recolectados:

- Campo obligatorio (*required*)
- Longitud máxima para textos
- Rango válido para números
- Validaciones de formato en emails y teléfonos (incluyendo validación por país)
- Tipado estricto de los *inputs*

Figura 8: Validacion de campo obligatorio.

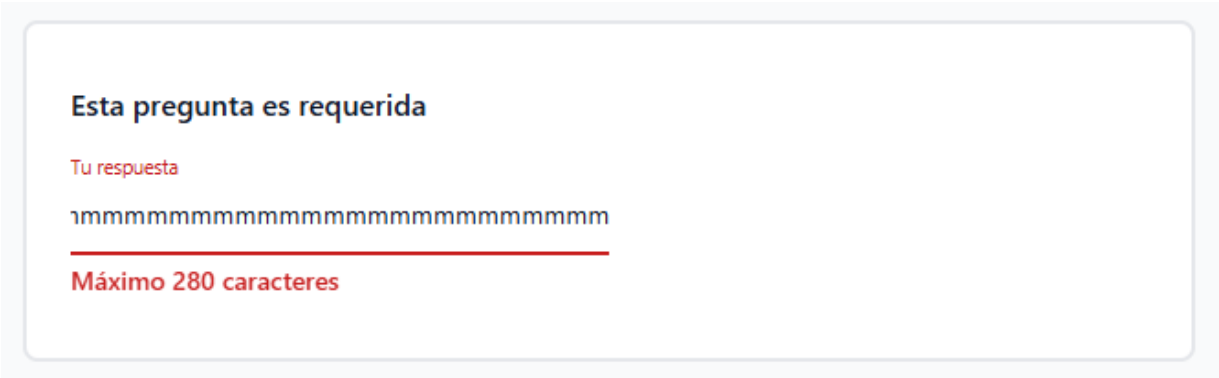

Figura 9: Validacion caracteres maximos

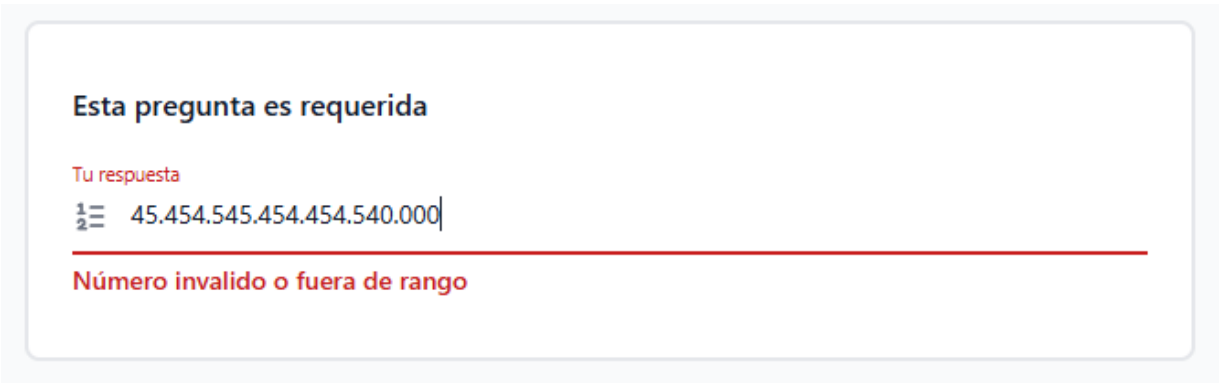

Figura 10: Validacion de rango en input numerico.

Figura 11: Validacion formade de email y telefono

Estas validaciones se aplican desde el frontend, asegurando que la información ingresada sea consistente y útil desde su origen.

El desarrollo del constructor se realizó dentro del dashboard **Console**, incorporando las vistas, rutas, componentes y *store* requeridos. Las tecnologías utilizadas fueron:

- **Vue.js 3** como *framework* principal [3].
- **TypeScript** para garantizar un tipado fuerte y mantenibilidad del código [6].
- **Flowbite** y **Tailwind CSS** para la construcción visual con componentes reutilizables [4, 5].
- **Pinia** para el manejo de estado global [15].

El diseño visual fue proporcionado por el equipo de UI/UX **Andares**, quienes elaboraron los componentes en *Figma*. El enfoque fue lograr una experiencia de usuario intuitiva y alineada con el resto del ecosistema de Truora.

Un aspecto importante fue la reutilización de componentes entre los dashboards *Console* y *Process-Runner*. Para esto, se creó un paquete compartido que evitó la duplicación de código, aunque implicó resolver conflictos de estilos y diferencias entre configuraciones de cada entorno.

La estructura del formulario se almacena como un objeto **JSON** en el frontend, el cual es enviado al backend donde se transforma en estructuras de **Go** y se persiste en **DynamoDB**. Adicionalmente, se indexa en **Elasticsearch** para facilitar búsquedas por campos como ID, nombre y contenido del formulario.

Durante el desarrollo surgieron varios retos, entre ellos:

- **Errores en *drag and drop*:** la lógica de reordenamiento y traslado de preguntas fue compleja y requirió varios ajustes para mantener la integridad del estado.
  Solución: se abordó el problema revisando más a fondo la documentación de la librería utilizada y realizando múltiples pruebas controladas hasta lograr un comportamiento estable y consistente.

- **Validación y mensajes de error:** se mejoró la experiencia del usuario con mensajes claros y precisos ante fallos de validación.
  Solución: originalmente se mostraban mensajes genéricos, por lo que se implementaron mensajes personalizados dependiendo del tipo de error, permitiendo al usuario identificar mejor el problema.

- **Store poco optimizada:** se identificaron funciones repetidas y se refactorizaron tras revisiones de código.
  Solución: se revisó detalladamente el store para detectar funciones redundantes, se centralizó la lógica en funciones reutilizables y se actualizaron los componentes para que utilizaran estas nuevas funciones optimizadas.

- **Problemas de autenticación en dashboard nuevo:** se desarrolló un sistema de *token refresh* para evitar cierres inesperados de sesión.
  Solución: el nuevo dashboard no renovaba automáticamente el token, provocando cierres abruptos. Se implementó un sistema de renovación automática del token, manteniendo la sesión activa mientras el usuario interactuaba con la aplicación.

- **Estilos conflictivos:** la reutilización de componentes entre proyectos generó conflictos de estilos, resueltos mediante ajustes específicos por entorno.
  Solución: el conflicto surgió porque un dashboard usaba Tailwind CSS y el otro no. Se agregaron configuraciones específicas en el dashboard sin Tailwind para garantizar la correcta visualización de los componentes compartidos, evitando así interferencias de estilo.

8.3.3. Almacenamiento y consulta de respuestas

Un aspecto esencial del proyecto fue diseñar un flujo sólido para capturar, almacenar y consultar las respuestas de los usuarios, permitiendo a los clientes de Truora obtener *insights* valiosos, hacer seguimiento y generar reportes personalizados.

Cuando un formulario está presente dentro de un flujo, el ID del formulario se incluye en la configuración del paso correspondiente. Durante la ejecución del flujo:

- El frontend solicita al backend la versión más reciente del formulario.
- El formulario se renderiza sección por sección.
- Las respuestas se envían al backend conforme el usuario avanza.
- Si las validaciones se cumplen, se activa la siguiente sección.
- Las respuestas se guardan con estado *pending*, permitiendo edición.
- Al finalizar todas las secciones, el formulario pasa a estado *finished*, permitiendo continuar con el flujo.

Las respuestas pueden visualizarse desde dos secciones principales del dashboard:

- En los resultados de procesos de identidad (vinculados a flujos).
- En una vista independiente desde la sección de formularios.

Estas vistas permiten a los clientes revisar datos registrados, hacer filtros personalizados y exportar resultados según sus necesidades operativas.

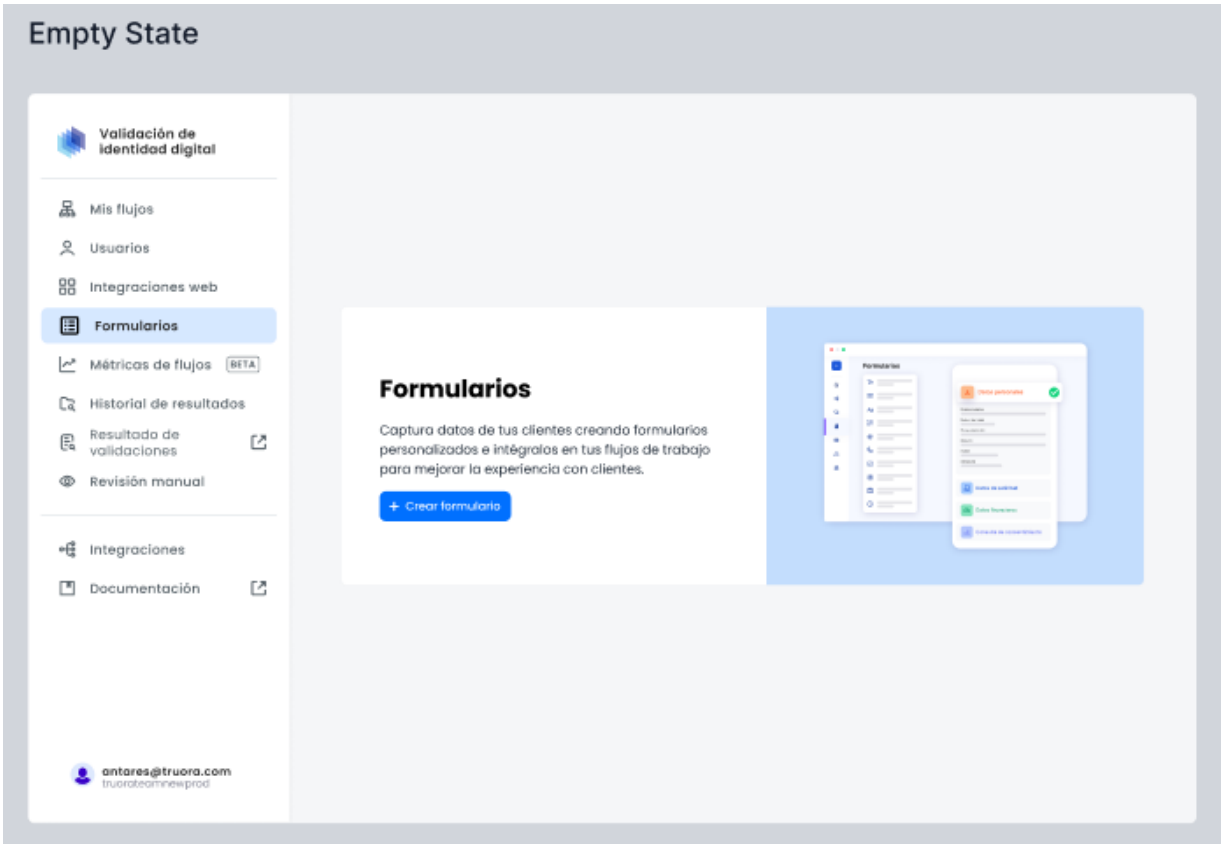

Figura 12: Vista inicial de la tabla de formularios sin registros creados. Se presenta un *empty state* que invita al usuario a crear su primer formulario mediante un botón de acción destacado.

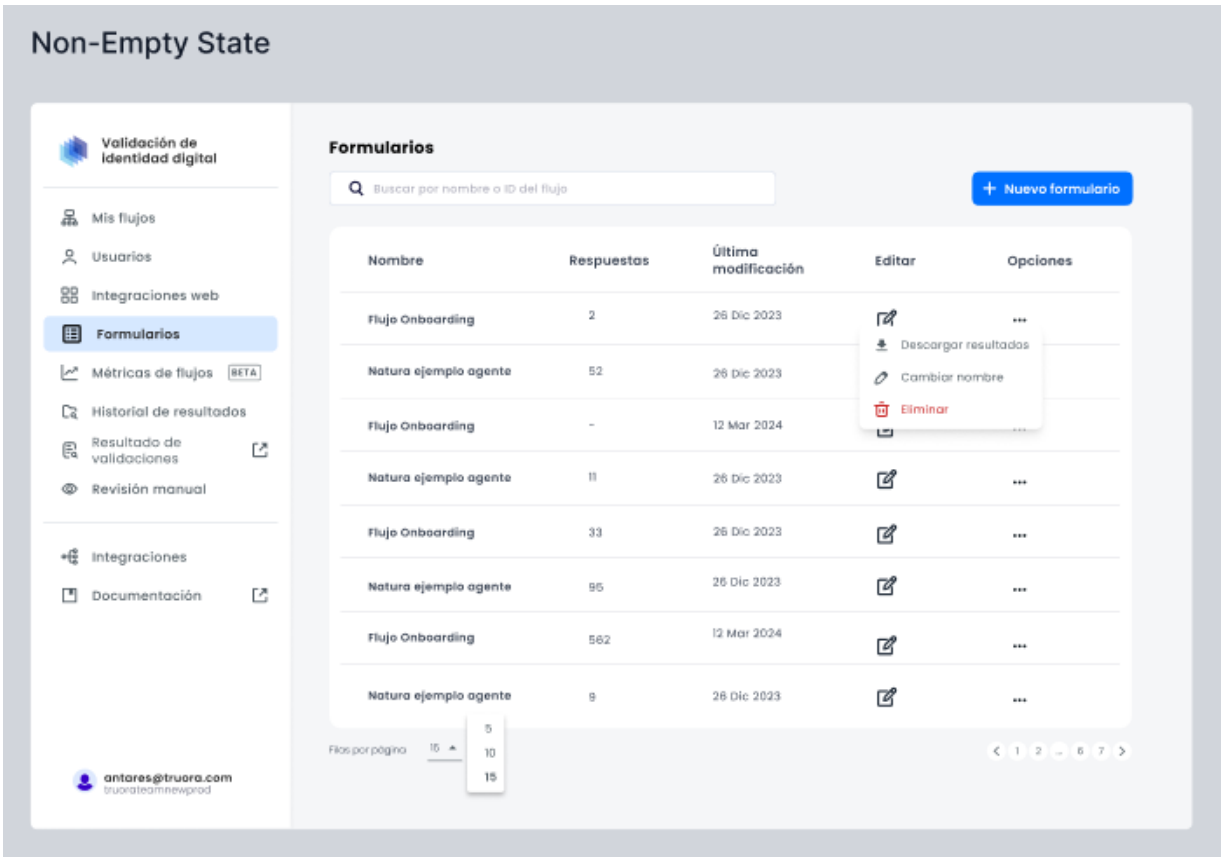

Figura 13: Tabla de formularios con registros existentes. El usuario puede buscar formularios por ID o nombre, y realizar acciones como eliminar, renombrar, descargar resultados, diligenciar el formulario o revisar su fecha de modificación.

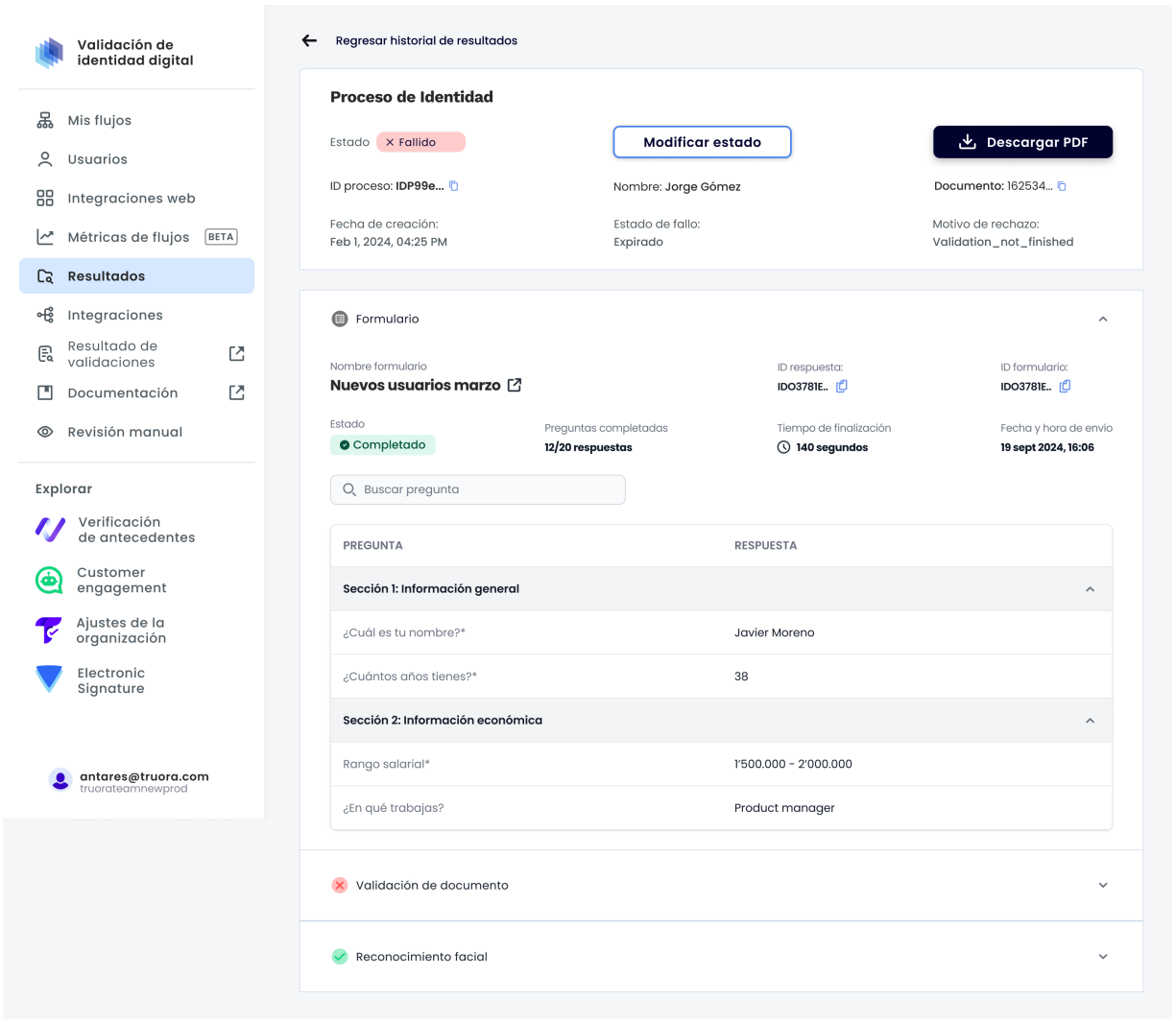

Figura 14: Visualización de respuestas dentro de un flujo de validación. Se muestran las secciones completadas, el tiempo de respuesta total y el detalle de cada sección respondida por el usuario.

8.4. Validar el funcionamiento del formulario web y su integración con los sistemas de Truora

Una vez completado el desarrollo del formulario web, se llevó a cabo un proceso exhaustivo de validación con el objetivo de asegurar tanto su correcto funcionamiento individual como su integración efectiva con los sistemas existentes de Truora. La validación se abordó desde múltiples frentes.

En la interfaz de usuario, se implementaron pruebas unitarias para verificar que cada tipo de campo de entrada (por ejemplo, texto, fechas, correos electrónicos, listas, entre otros) cumpliera con las validaciones correspondientes. Además, se organizaron sesiones de pruebas manuales en colaboración con el equipo de diseño (el equipo Antares), en las que se evaluaron los distintos componentes del formulario en un entorno de uso real. Estas sesiones permitieron identificar y corregir errores visuales, inconsistencias en las validaciones y detalles relacionados con márgenes, tamaños y espaciado.

Por el lado del servidor, se desarrollaron pruebas unitarias utilizando el lenguaje Go [1] con el fin de asegurar que los datos enviados desde el formulario fueran recibidos en un formato válido antes de ser almacenados en la base de datos DynamoDB [7]. También se implementaron pruebas funcionales automatizadas sobre los puntos de acceso más críticos, las cuales se ejecutan periódicamente en un entorno alterno para detectar fallos o comportamientos inesperados.

La validación de la integración con los flujos de identidad se realizó mediante pruebas completas, que incluyeron desde la creación del formulario y su uso dentro de un flujo, hasta la visualización y descarga de respuestas. Estas pruebas garantizaron que los formularios se comportaran adecuadamente dentro del proceso de identidad, respetando la lógica de pasos y permitiendo el envío progresivo de secciones.

Adicionalmente, se contó con el apoyo del equipo de pruebas funcionales (el equipo Hamal), encargado de validar los productos mediante casos de uso reales. Su retroalimentación fue clave para mejorar aspectos funcionales del formulario. Finalmente, se elaboró documentación interna dirigida a desarrolladores sobre los puntos de acceso del sistema, así como guías públicas orientadas a los usuarios finales sobre cómo crear formularios e integrarlos dentro de los flujos de identidad.

8.5. Desplegar y monitorear el constructor de formularios web en el entorno de produccion asegurando el correcto funcionamiento con todos los servicios

El constructor de formularios web fue desplegado utilizando la infraestructura en la nube de Amazon Web Services (AWS), el entorno estándar de ejecución y despliegue en la empresa Truora [2]. A diferencia de otros entornos con integración continua, en Truora el despliegue se lleva a cabo de forma manual, aunque apoyado por agentes internos que facilitan la ejecución del proceso.

El flujo de despliegue consiste en ejecutar comandos específicos para compilar (*build*) el proyecto y luego desplegarlo en el entorno deseado. Estos comandos hacen referencia al nombre del proyecto o a funciones `AWS Lambda` concretas [9]. Para la gestión y provisión de la infraestructura, se utiliza la herramienta Terraform, que permite definir la infraestructura como código (*Infrastructure as Code, IaC*) [16]. Esto facilita la replicabilidad, mantenibilidad y trazabilidad de los recursos desplegados.

Dentro de la organización se definen dos entornos principales: **staging** (preproducción) y **producción**, cada uno asociado con comandos de despliegue particulares. Dependiendo de la criticidad de los cambios realizados o los recursos involucrados, algunos despliegues requieren permisos especiales. En esos casos, los planes generados por Terraform deben ser revisados y aprobados por usuarios con roles de *admin* o *super admin*, definidos por los líderes técnicos del equipo. Esta capa adicional de control permite asegurar un mayor grado de gobernanza y protección ante posibles fallos en producción.

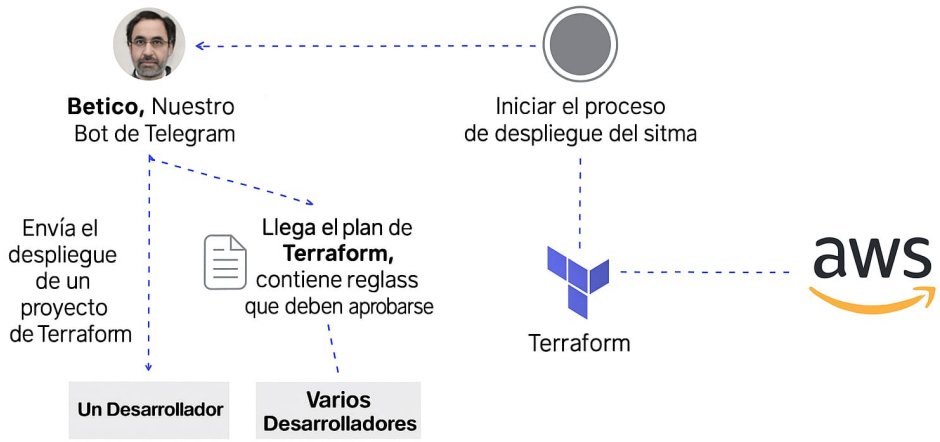

Figura 15: Diagrama de flujo despliegues

9. Desarrollo y ejecución de pruebas

El presente capítulo describe el proceso de desarrollo y ejecución de un conjunto integral de pruebas diseñado para validar que todos los requisitos definidos en las etapas iniciales del proyecto han sido correctamente implementados en el software.

El enfoque de pruebas adoptado busca garantizar tanto la calidad del código como la estabilidad funcional del sistema. Para ello, se estableció como objetivo mínimo alcanzar un 80 % de cobertura de código en las pruebas automatizadas, asegurando así que una proporción significativa del código sea verificada de manera sistemática.

Las pruebas abarcaron diferentes niveles y componentes del sistema:

- **Pruebas unitarias en el backend**, escritas en Go, con un umbral de cobertura superior al 80 %.
- **Pruebas funcionales y pruebas end-to-end (E2E)** mediante Cypress, orientadas a validar los flujos completos de usuario y la integración de los distintos módulos.
- **Pruebas automáticas sobre los servicios expuestos como endpoints** (lambdas integradas con API Gateway), que se ejecutan automáticamente en cada despliegue del proyecto de formularios.
- **Pruebas unitarias en el frontend**, enfocadas en componentes desarrollados en Vue 3 y en los *stores* del sistema. Estas pruebas garantizan la correcta funcionalidad aislada de cada elemento de la interfaz de usuario y también están sujetas a umbrales de cobertura preestablecidos.

El control de la cobertura de código es riguroso, tal como se refleja en los reportes generados durante el proceso de integración continua. Un ejemplo típico de reporte que evidencia el incumplimiento de las métricas es el siguiente:

```
ERROR: Coverage for lines (54.54%) does not meet global threshold
   (85%) for src/components/FormBuilderDynamicListConfig.vue
ERROR: Coverage for functions (0%) does not meet global threshold
   (80%) for src/components/FormBuilderDynamicListConfig.vue
ERROR: Coverage for statements (54.54%) does not meet global
   threshold (85%) for src/components/FormBuilderDynamicListConfig
   .vue
ERROR: Coverage for branches (0%) does not meet global threshold
   (80%) for src/components/FormBuilderDynamicListConfig.vue
```

Estos resultados permiten identificar de manera precisa los componentes que requieren refuerzo en las pruebas automatizadas, garantizando la mejora continua de la calidad del software.

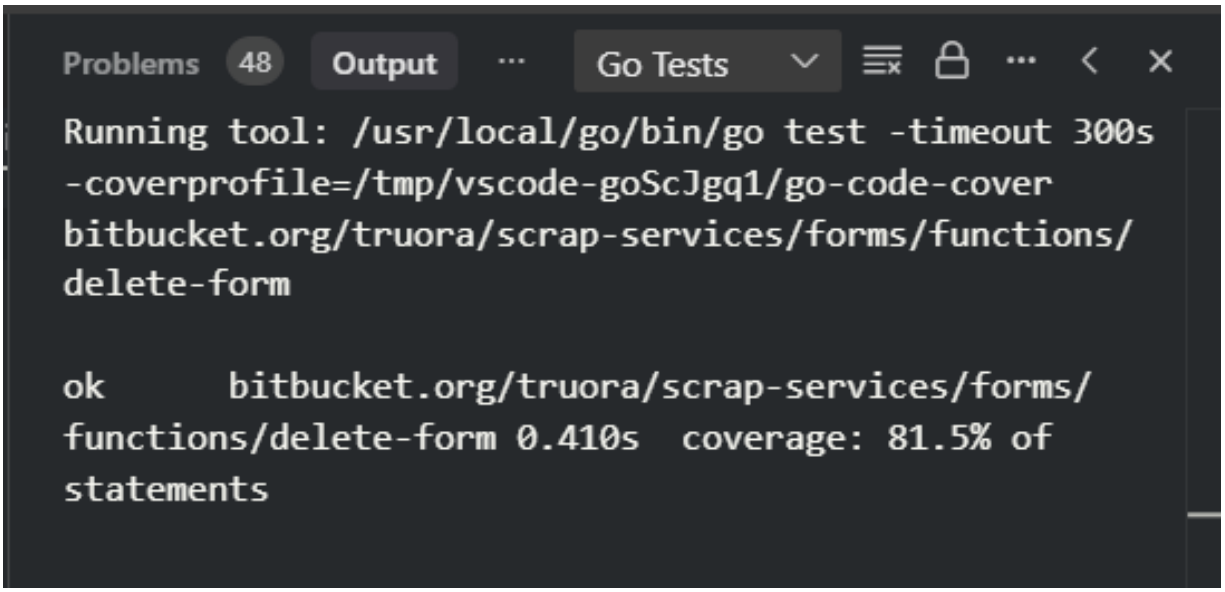

Figura 16: Coverage en test de go

Algorithm 1: Test funcional: Crear formulario exitosamente

Input: Formulario de prueba `mockForm`, URL de la API, API Key
Output: Formulario creado y eliminado exitosamente

1 Inicializar validaciones `c`;
2 Serializar `mockForm` a JSON `createFormPayload`;
3 Verificar que no haya error en la serialización;
4 Enviar petición POST a `/forms/` con `createFormPayload` y guardar en `createdForm`;
5 Verificar que no haya error en el POST;
6 Verificar que el código HTTP sea 200 OK;
7 Verificar que `createdForm.Sections` tiene longitud 2;
8 Verificar que `createdForm.Sections[0].Inputs` tiene longitud 5;
9 Verificar que `createdForm.Sections[1].Inputs` tiene longitud 4;
10 Verificar que `createdForm.Version` es igual a 1;
11 Enviar DELETE a `/forms/{formID}` con la API Key;
12 Verificar que no haya error en el DELETE;
13 Verificar que el código HTTP sea 200 OK;

Algorithm 2: Test unitario: Renderizar formulario por defecto

Input: Componente `FormBuilder`, plugins `pinia`, `router`, función mock `$t`
Output: Formulario renderizado correctamente con componentes esperados

1 Montar el componente `FormBuilder` con los siguientes parámetros:;
2 Plugins: `pinia`, `router`;
3 Mocks: función de traducción `$t(key: string) → key`;
4 Esperar el siguiente ciclo del DOM usando `$nextTick`;
5 Verificar que `mixpanel.track` fue llamado correctamente;
6 Verificar que el nombre del formulario sea `"Mi primer formulario"`;
7 Verificar que la longitud de las secciones del formulario sea igual a 1;
8 Verificar que se rendericen 2 componentes del tipo `FormBuilderQuestionCard`;
9 Verificar que no se rendericen componentes del tipo `FwbBadge`;

10. Marco de referencia

10.1. Metodología de desarrollo

En el presente proyecto se adoptará la metodología Scrumban, la cual integra principios de Scrum y Kanban, con el objetivo de asegurar un enfoque ágil y eficiente en la gestión del desarrollo. La administración del proyecto se llevará a cabo mediante la plataforma de Atlassian, aprovechando la integración de herramientas como Bitbucket, destinada al alojamiento de las diferentes versiones del código, y Jira, utilizada para el seguimiento detallado de actividades, tareas pendientes e incidencias.

La comunicación con el equipo de desarrollo se efectuará mediante canales de mensajería instantánea en Telegram, complementándose con reuniones virtuales periódicas a través de Zoom para facilitar la coordinación y revisión de avances.

Como parte integral del proyecto, se desarrollará una investigación exhaustiva sobre herramientas, metodologías y casos de uso reales aplicables a la plataforma AWS, conforme a los lineamientos establecidos por la organización. Esta etapa tiene como finalidad identificar soluciones óptimas que satisfagan los requerimientos actuales del sistema en desarrollo.

Para la implementación técnica, se empleará el lenguaje de programación Go en el desarrollo del backend, mientras que en el frontend se utilizarán tecnologías como Vue.js, Flowbite y TailwindCSS, maximizando el uso de los recursos provistos por los servicios en la nube de AWS.

El equipo de trabajo estará conformado por tres desarrolladores y un gerente de producto, quienes colaborarán estrechamente con el pasante a través de sesiones de pair programming y revisiones de código. Además, se realizarán presentaciones periódicas de avances técnicos ante el gerente de producto para su evaluación y control.

En cuanto a la seguridad del servicio, esta será implementada en conformidad con los lineamientos de Truora, garantizando el cumplimiento de las normativas de privacidad estipuladas por la norma ISO/IEC 27001 [17].

Finalmente, se ejecutarán pruebas en cada etapa del desarrollo conforme a un plan de pruebas previamente definido, y se establecerá un monitoreo continuo del servicio en producción, con el fin de asegurar su correcto funcionamiento y un rendimiento óptimo.

10.2. Tecnologías de implementación

Go

Go es un lenguaje de programación de código abierto que facilita la creación de software simple, confiable y eficiente [1].

Amazon Web Services (AWS)
Plataforma de servicios web de infraestructura en la nube dirigida a organizaciones de todos los tamaños [2].

AWS Lambda
Servicio web que permite ejecutar código sin necesidad de aprovisionar ni administrar servidores. Es posible configurar su ejecución automática en respuesta a eventos de otros servicios de AWS o mediante llamadas directas desde aplicaciones web o móviles [9].

Amazon API Gateway
Servicio completamente administrado que facilita la creación, publicación, mantenimiento, monitoreo y protección de APIs a cualquier escala [8].

Amazon Kinesis
Plataforma diseñada para el procesamiento de datos en tiempo real mediante transmisión (*streaming*) dentro de AWS. Ofrece servicios que simplifican la carga y análisis de dichos datos [18].

Amazon CloudWatch Events
Servicio web que permite generar flujos de eventos del sistema en tiempo real, notificando cambios en los recursos de AWS a servicios como Lambda, Kinesis o SNS [14].

Amazon ECR
Amazon Elastic Container Registry (ECR) es un registro de contenedores Docker completamente administrado, integrado con Amazon ECS e IAM, que permite almacenar, gestionar y desplegar imágenes de contenedores [19].

Amazon DynamoDB
Base de datos NoSQL completamente administrada, que proporciona un rendimiento rápido, predecible y altamente escalable [7].

Vue.js
Framework progresivo de JavaScript para la construcción de interfaces de usuario, basado en HTML, CSS y JavaScript, con un enfoque declarativo y orientado a componentes [3].

Flowbite
Biblioteca de componentes de interfaz de usuario de código abierto basada en Tailwind CSS. Incluye soporte para modo oscuro, plantillas prediseñadas y un sistema de diseño visual en Figma [4].

Tailwind CSS
Framework de utilidades CSS que permite diseñar interfaces directamente desde las clases del HTML, evitando estilos predefinidos para componentes como botones o tablas [5].

TypeScript
Lenguaje de programación fuertemente tipado que se basa en JavaScript y proporciona herramientas mejoradas para el desarrollo a gran escala [6].

11. Conclusiones

El desarrollo del constructor de formularios web, en el contexto de la pasantía realizada en la empresa Truora, representó una experiencia formativa de alto valor, tanto en términos técnicos como organizacionales. A lo largo del proyecto se profundizó en prácticas de desarrollo de productos digitales, al tiempo que se obtuvo una comprensión integral del funcionamiento interno de una compañía tecnológica en constante evolución.

Uno de los aprendizajes más relevantes fue el entendimiento de cómo las decisiones tecnológicas se vinculan directamente con los objetivos estratégicos del negocio. La implementación de un servicio como el constructor de formularios se evidenció como un elemento diferenciador dentro del mercado, generando valor tanto para la organización como para sus usuarios finales. En este proceso se destacó la relevancia de diseñar soluciones centradas en el usuario, lo que fortalece las relaciones comerciales y facilita la creación de nuevas oportunidades de negocio.

La experiencia también permitió valorar la importancia de la colaboración interdisciplinaria. La interacción continua con los equipos de producto, tecnología y negocio evidenció cómo una comunicación efectiva y un trabajo en equipo bien coordinado son fundamentales para el éxito de los proyectos. Asimismo, se vivenció la dinámica de un entorno ágil, adaptativo y orientado a resultados, donde la mejora continua es una constante.

Desde la perspectiva técnica, el aporte principal consistió en el diseño e implementación de un componente estratégico para la plataforma de Truora. El constructor de formularios optimizó procesos clave como el onboarding y la validación de identidad, centralizando la captura de datos y eliminando la dependencia de proveedores externos. Esto fortaleció el posicionamiento de la empresa como proveedor integral.

El proyecto abordó todas las etapas del ciclo de vida de desarrollo de software: desde el levantamiento de requerimientos, con la definición de funcionalidades clave (creación y ordenamiento de campos, validaciones dinámicas, etc.), hasta la arquitectura de la solución sobre los servicios implementados.

Se desarrolló además un sistema robusto de almacenamiento y consulta de respuestas, integrado al dashboard de Truora y conectado al motor de reglas de negocio (BRE). Las pruebas abarcaron distintos niveles (unitarias, funcionales y end-to-end con **Cypress**) y permitieron garantizar la calidad del producto final. Finalmente, la puesta en producción, asegurando un despliegue estable, escalable y monitoreado dentro del entorno de AWS [2].

En conjunto, esta experiencia representó una oportunidad valiosa para aplicar y expandir conocimientos en ingeniería de software, arquitectura de sistemas en la nube y metodologías ágiles dentro de un contexto empresarial real y de alto impacto.

12. Trabajos Futuros

Tras el desarrollo inicial del constructor de formularios web y su integración en los procesos de *onboarding* de la plataforma de Truora, se identifican nuevas oportunidades para seguir ampliando el alcance y el impacto de esta solución dentro del ecosistema de productos de la compañía.

El siguiente paso estratégico será extender esta funcionalidad al entorno de **WhatsApp**, aprovechando las capacidades de **WhatsApp Flows**. La integración de formularios interactivos directamente en los flujos de conversación de WhatsApp permitirá a los clientes de Truora ofrecer experiencias de *onboarding* y captura de información aún más fluidas y accesibles para sus usuarios finales, en un canal ampliamente utilizado.

Este desarrollo no solo aportará valor en términos de experiencia del usuario, sino que también permitirá que la información recolectada a través de los formularios en WhatsApp se integre de manera nativa con nuestro producto de **Customer Engagement**. De este modo, los datos obtenidos podrán ser utilizados como propiedades de contacto enriquecidas, habilitando segmentaciones más precisas, personalización de comunicaciones y un mejor seguimiento a lo largo del ciclo de vida del cliente.

Entre los principales objetivos de este trabajo futuro se destacan:

- **Diseñar e implementar formularios** adaptados a la experiencia conversacional en WhatsApp, aprovechando las capacidades de WhatsApp Flows.
- **Integrar los datos capturados** en estos formularios con el motor de Customer Engagement, para que sean utilizados como propiedades de contacto en campañas y automatizaciones.
- **Optimizar la experiencia de usuario** en flujos de WhatsApp, eliminando fricciones y mejorando la conversión y la calidad de la información recolectada.
- **Consolidar aún más la visión de plataforma integral (*one platform*)** de Truora, ofreciendo a los clientes una solución completa que abarque múltiples canales de interacción y gestión de usuarios.

Este enfoque permitirá fortalecer la posición de Truora como un socio estratégico para sus clientes, ofreciendo capacidades avanzadas de captura y gestión de datos a través de los canales digitales más relevantes. A futuro, se continuará explorando nuevas formas de enriquecer el ecosistema de productos de la compañía, con el objetivo de ofrecer soluciones cada vez más integradas y centradas en el usuario.

13. Referencias

[1] "The Go Programming Language," Go Docs. [En línea]. Disponible: https://golang.org/doc/

[2] "What is AWS?," Amazon Web Services, Inc. [En línea]. Disponible: https://aws.amazon.com/what-is-aws/

[3] E. You, "Vue.js Guide," Vue.js. [En línea]. Disponible: https://vuejs.org/v2/guide/

[4] "Flowbite: Utility-first CSS-in-JS UI components built with Tailwind CSS and React," Flowbite. [En línea]. Disponible: https://flowbite.com/docs/getting-started/

[5] A. Medinets, "Tailwind CSS - Rapidly build modern websites without ever leaving your HTML," Tailwind Labs. [En línea]. Disponible: https://tailwindcss.com/docs

[6] "TypeScript - JavaScript that scales," TypeScript. [En línea]. Disponible: https://www.typescriptlang.org/docs/

[7] "Amazon DynamoDB," Amazon Web Services. [En línea]. Disponible: https://docs.aws.amazon.com/dynamodb/

[8] "Amazon API Gateway," Amazon Web Services. [En línea]. Disponible: https://docs.aws.amazon.com/apigateway/

[9] "AWS Lambda," Amazon Web Services. [En línea]. Disponible: https://docs.aws.amazon.com/lambda/

[10] "Amazon ECS (Elastic Container Service)," Amazon Web Services. [En línea]. Disponible: https://docs.aws.amazon.com/ecs/

[11] "Amazon S3," Amazon Web Services. [En línea]. Disponible: https://docs.aws.amazon.com/s3/

[12] "Amazon Simple Queue Service (SQS)," Amazon Web Services. [En línea]. Disponible: https://docs.aws.amazon.com/sqs/

[13] "Amazon CloudFront," Amazon Web Services. [En línea]. Disponible: https://docs.aws.amazon.com/cloudfront/

[14] "Amazon CloudWatch," Amazon Web Services. [En línea]. Disponible: https://docs.aws.amazon.com/cloudwatch/

[15] "Pinia - The intuitive store for Vue.js," Pinia Docs. [En línea]. Disponible: https://pinia.vuejs.org

[16] "Terraform by HashiCorp," Terraform Docs. [En línea]. Disponible: https://www.terraform.io/

[17] International Organization for Standardization. *ISO/IEC 27001:2022 - Information security, cybersecurity and privacy protection — Information security management systems — Requirements*. ISO, 2022.

[18] Amazon Web Services, Inc. *Amazon Kinesis Documentation*. Disponible en: `https://docs.aws.amazon.com/kinesis/`, consultado en julio de 2025.

[19] Amazon Web Services, Inc. *Amazon Elastic Container Registry (ECR) Documentation*. Disponible en: `https://docs.aws.amazon.com/AmazonECR/`, consultado en julio de 2025.