UNIVERSIDAD DE ANTIOQUIA

DESARROLLO FRONT-END DE UNA APLICACIÓN EMPRESARIAL PARA LA GESTIÓN DE ÓRDENES.

Autor(es)

Manuel Alejandro Gómez Ortega

Universidad de Antioquia

Facultad de Ingeniería

Departamento de Ingeniería de Sistemas

Medellín, Colombia

2021

DESARROLLO FRONT-END DE UNA APLICACIÓN EMPRESARIAL PARA LA GESTIÓN DE ÓRDENES.

Informe práctica empresarial como requisito parcial
para optar al título de Ingeniero de Sistemas

Autor

Manuel Alejandro Gómez Ortega

Asesor interno

Luis Hernando Silva Florez

Asesor externo

Pablo Alejandro Giorgi

Universidad de Antioquia
Facultad de Ingeniería

Departamento de Ingeniería de Sistemas
Medellíın, Colombia

2021

Índice

1. Resumen

Un importante cliente internacional de la empresa de software **WOLOX** definió dentro de su estrategia de transformación digital la necesidad de llevar los procesos que se realizaban en mayor parte de forma manual a lo digital, transformando así la atención y mejorando la experiencia de sus clientes. Para ayudar a resolver esta necesidad se planteó el desarrollo de una aplicación web que permitiera gestionar las órdenes y clientes. Este proyecto en específico describe cómo se desarrolló esta aplicación desde la capa **frontend**, detallando algunos de los componentes que pudieron aportar al cumplimiento de las funcionalidades que requería el cliente para lograr transformar sus procesos. El resultado es una aplicación web desarrollada utilizando la librería React que cuenta con cuatro niveles de autenticación (roles) y permite listar y crear órdenes, comunicándose por medio de REST a varios servicios que brindan la información y las funciones de almacenamiento de datos. Dentro de esta aplicación se utilizaron diversas librerías que facilitan cumplir con el objetivo planteado. Al final de este proyecto se pudo obtener una aplicación que contempla los módulos de autenticación y órdenes, ambos módulos pudieron ser probados tanto por el equipo interno de calidad como por el mismo cliente a través de terceros que velaban por la seguridad de la aplicación y también por algunos stakeholders que pudieron experimentar la aplicación en un ambiente productivo.

Palabras clave: aplicación web, React, frontend, rest, autenticación.

2. Introducción

La compañía **WOLOX** es una empresa dedicada al desarrollo de software con presencia a nivel internacional, cuenta con oficinas en Estados Unidos y varios países de latinoamérica, con más de 400 empleados y referentes tecnológicos importantes en la industria. Dado que el proyecto trabajado contaba con un contrato de confidencialidad existieron detalles sobre la aplicación que han sido omitidos en pro de respetar este contrato. Este proyecto hacía parte de una estrategia de transformación digital de una importante empresa internacional (en adelante denominada **EL CLIENTE**) a la cual Wolox, en esta oportunidad, le brindó sus servicios de desarrollo de software.

Dentro de las necesidades del cliente se encontraba llevar los procesos que se realizaban en mayor parte de forma manual a lo digital, transformando así la atención y mejorando la experiencia de sus clientes, dada esta necesidad se planteó una solución que corresponde a una plataforma web donde sus tres entidades (clientes, agentes y administradores) pudiesen interactuar y llevar a cabo los procesos de forma sistematizada.

En este proyecto en específico se trató únicamente la capa front-end, es decir, *"un programa que es destinado para el uso por parte de usuarios finales. Suelen contar con una interfaz gráfica y requieren de un servidor o Backend para funcionar."* **[1]** y se trabajó usando la librería React del cual hablamos más adelante. Wolox es una empresa que cree en las metodologías ágiles y este proyecto no estuvo exento de ellas, se trabajó usando la metodología **SCRUM** adaptado bajo los estándares de Wolox, adicionalmente este proyecto contó con un equipo robusto de profesionales, los cuales complementaron cada una de las áreas que deben ser necesarios para hacer que el proyecto pudiera funcionar: diseño, pruebas, contabilidad, gestión de proyecto y por supuesto desarrollo. Al haber sido este proyecto una etapa inicial de un proceso de transformación digital del cliente, se planteó que esta solución debería tener como entrega un **MVP** (mínimo producto viable o como lo indican sus siglas en inglés minimum viable product).

3. Objetivos

Objetivo general

Desarrollar varios componentes de una aplicación frontend en React con autenticación y roles, para gestionar órdenes de una empresa, siguiendo unos diseños UI previamente desarrollados por parte del equipo y cumpliendo con los criterios de aceptación para ser aprobados por el equipo de QA.

Objetivos específicos

- Desarrollar las vistas basadas en los diseños provistos por el equipo de UI necesarias para el correcto funcionamiento de los módulos de la aplicación.
- Desarrollar un componente de autenticación que permita identificar un usuario por medio de su usuario y contraseña.
- Construir un módulo que ofrezca al usuario opciones para interactuar con las órdenes almacenadas en la base de datos a través de un servicio provisto por el backend.

4. Marco teórico

Es clave iniciar definiendo el concepto de FrontEnd, como lo mencionamos anteriormente es la capa de la aplicación que se encarga de mostrarle al usuario una interfaz amigable para que este interactúe con ella. Esta capa no debe confundirse con el diseño de la aplicación dado que el FrontEnd no es únicamente una representación gráfica de unas acciones sino que también tiene su propia lógica interna para reaccionar a diferentes situaciones dependiendo de la interacción que el usuario realice con este.

Javascript

Una vez entendido que es el frontend podemos repasar sobre el lenguaje a trabajar en esta ocasión, este puntualmente es llamado Javascript (JS por su abreviatura) *"JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario"* **[2]**.

Auth0

La seguridad en una aplicación web es primordial, tener un método seguro de autenticación que provea herramientas para validar al usuario y prevenir suplantación de identidad es una característica que debe llevar por defecto cualquier aplicación empresarial. En esta ocasión hemos usado Auth0 que es una plataforma que ofrece su servicio de autenticación con un paquete para desarrolladores con múltiples funciones y servicios que permiten hacer una integración segura y sencilla en pocos pasos.

React

Existen múltiples marcos de trabajo basados en Javascript, sin embargo dada la experiencia de Wolox y las negociaciones realizadas con el cliente, se utilizará la librería React.js para este proyecto; *"React, también conocido como React.js o ReactJS, es una librería JavaScript OpenSource diseñada para crear interfaces de usuario. Ha sido concebida para facilitar el desarrollo de SPA, Single Page Applications, obteniendo un gran rendimiento y ofreciendo una forma de desarrollo más cercana a la creación de videojuegos que a la de*

aplicaciones. Esta librería está mantenida por Facebook, Instagram y una gran comunidad de desarrolladores independientes y corporaciones". **[3]** A lo largo del desarrollo nos encontramos con que podremos usar una o varias librerías adicionales que permitirán ahorrar tiempo de desarrollo y brindar mayores funcionalidades a la aplicación.

React Hook Form

Siguiendo las últimas tendencias en desarrollo se escogió la librería React Hook Form como nuestra principal herramienta para desarrollar los formularios que se muestran dentro de la aplicación, esta librería contiene un paquete de funciones que permiten hacer validaciones e integrar campos de texto personalizados, lo que nos servirá para las diferentes vistas de creación de orden y autenticación del usuario.

Control de versiones (GIT)

Finalmente, otro de los conceptos a utilizar en el proyecto, y de gran importancia debido a que se dispone de un equipo con un número considerable de participantes, es el control de versionamiento de código, esto es, tener un registro y un lugar donde se almacene el código que cada uno de los miembros del equipo de desarrollo finalice para luego unir todas las versiones del código en una sóla, de tal forma que se garantice que esté organizado y listo para ser ejecutado. Para esto utilizaremos Git, un sistema de control de versiones que permite gestionar el cambio **[4],** sincronizar el trabajo entre cada cambio por cada miembro del equipo y poder mantener un registro de quién, cuándo y en donde se ha modificado el código de la aplicación.

Testing

Al ser una aplicación empresarial siempre debe estar considerado que la aplicación tenga pruebas, las pruebas son vitales dentro de una aplicación ya que se verifica que ninguna de las nuevas implementaciones generen errores en funcionalidades existentes **[5]**, adicionalmente garantizamos que estas tengan los comportamientos esperados en cada interacción con los servicios o con el usuario, en esta ocasión utilizamos la librería **React Testing Library** para realizar las pruebas de la aplicación y se buscó llegar a un cubrimiento del código mayor al 80%, esto es una medida generada por la herramienta de pruebas Jest**[6]** que mide según nuestras líneas de código cuántas cuentan con pruebas.

5. Metodología

En **Wolox** todos los proyectos son desarrollados bajo las metodologías ágiles, más específicamente en este caso con el marco de trabajo **SCRUM**, es "una metodología ágil para la gestión de proyectos relacionados con la construcción (desarrollo) de Software." **[7]** su principal objetivo es realizar iteraciones con entregables que permitan al cliente ir visualizando elementos tangibles del desarrollo e identificar posibles errores en la marcha evitando reprocesos en momentos ya muy avanzados que impliquen altos costos de tiempo/dinero cómo puede llegar a pasar en otras metodologías donde sólo se pueden ver estos resultados al final sin posibilidad de estar iterando.

SCRUM

Para llevar a cabo cada uno de los pasos del scrum que detallamos a continuación utilizamos la herramienta **JIRA** una plataforma de planificación de procesos que está diseñada para llevar a cabo los procesos que hacen parte de **SCRUM** de una manera organizada y amigable con el usuario. A continuación vemos una gráfica de cada uno de los pasos y ceremonias que hacen parte de **SCRUM**:

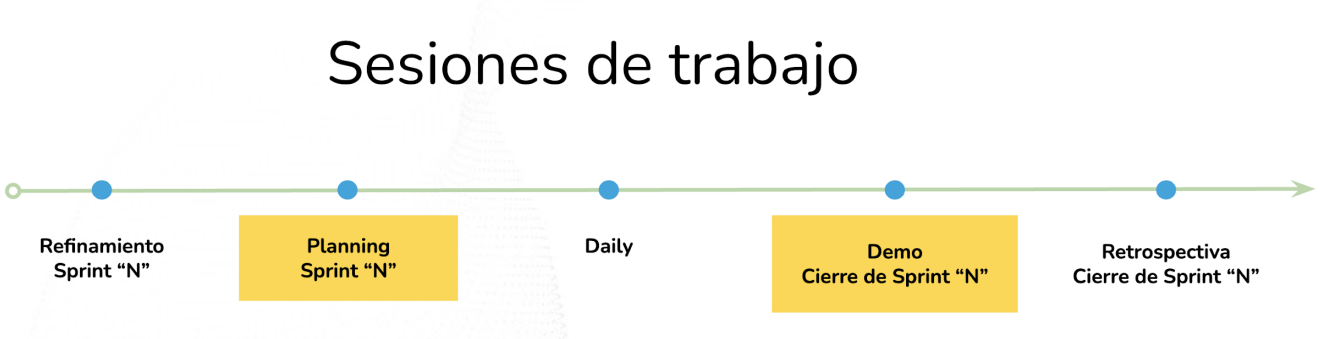

Figura 1. Sesiones de trabajo SCRUM

Refinamiento: Preparación y especificación de tareas del siguiente sprint
Planning: El objetivo de esta reunión es alinear y consensuar las tareas que se van a hacer en el sprint
Daily: Cada Sprint está compuesto por ciclos diarios que comienzan con las daily meetings para tener seguimiento del "día a día".
Demo: Instancia final de cada sprint, en la cual el equipo asignado al proyecto expone lo avanzado.

Retrospectiva: Análisis de los problemas que puedan haberse dado en la ejecución del proyecto, la revisión de las velocidades del equipo, así como la revisión y aclaración de objetivos internos.

Hicieron parte del equipo un **Arquitecto** que se encarga de diseñar toda la infraestructura de la aplicación, el equipo de **diseño UI/UX** que se encargaron de desarrollar las vistas de la aplicación de forma creativa y amigable para el usuario, un **Team Manager** que se encargó de velar por que el proyecto saliera adelante y gestionar todas las tareas y tiempos de cada uno de los miembros del equipo, una gerente de cuenta encargada de las negociaciones con el cliente, un equipo de QA que veló por que la aplicación cumpla sus funciones de manera correcta y que se cumplieran los criterios de aceptación de cada una de las funcionalidades, un equipo desarrollo dividido en front y back y finalmente una serie de stakeholders por parte del cliente que velaron por que se cumplan los objetivos y colaboran de manera activa cuando se requiera una información, herramienta o acceso para continuar con el desarrollo.

Conforme pasó el tiempo del proyecto, en las últimas iteraciones, el equipo por parte de Wolox se fue haciendo más pequeño, dado que en las últimas instancias del proyecto ya el equipo de arquitectura y diseño ui/ux había culminado su trabajo por lo cual fueron asignados a otros proyectos/responsabilidades por fuera de este proyecto. Esto a su vez fortaleció el vínculo entre los desarrolladores y el cliente, dado que este tuvo que estar más fuertemente involucrado en las decisiones de diseño finales, para esto tuvimos la oportunidad de contar con una persona haciendo el rol de **Product Owner[8]** por parte del cliente.

Estructura del proyecto

En cuanto al desarrollo nos enfrentamos a un proyecto que fue hecho utilizando la tecnología React y que tiene una estructura de carpetas partiendo de lo que nos ofrece el **create-react-app,** la cual detallamos a continuación:

- Carpeta raíz: **NOMBRE-PROYECTO-FRONTEND-WEB**
  - **node_modules:** aquí se descargan y extraen todas las librerías/paquetes utilizados en el proyecto.
  - **build:** cuando el proyecto se exporta para producción en esta carpeta quedan los archivos necesarios para su ejecución.

- **public:** en esta carpeta está la definición de los archivos expuestos al público como imágenes, manifest, favicon y demás archivos estáticos que necesite la aplicación por fuera de React.
- **src:** dentro de esta carpeta se encuentra la fuente del código completa de toda la aplicación y se estructura así:
  - **assets/**: imágenes o iconos necesarios para ser llamados dentro de los componentes.
  - **components/:** componentes que pueden reutilizarse en toda la aplicación. Un componente puede tener a su vez componentes internos, los cuales deberán estar dentro de una carpeta llamada components que debe estar dentro del componente padre.
  - **config/:** archivos de configuración necesarios para las librerías o el proyecto en general.
  - **contexts/:** definición de los contextos a utilizar dentro del proyecto (Context API React)
  - **hooks/:** hooks para ser utilizados en toda la aplicación (ej: useRequest para hacer solicitudes asíncronas)
  - **mocks/:** objetos dummy para utilizar dentro de la aplicación cuando no se cuentan con datos desde la integración para hacer pruebas.
  - **screens/:** aquí estarán todas las vistas que requiera la aplicación, una vista se puede estructurar así:
    - **nombreDeLaScreen/**
      - **components/**: aquí estarán los componentes que se usan exclusivamente dentro de esta screen
      - **screens/:** aquí están las screens que se usarán exclusivamente dentro de esta screen
      - **index.tsx**
      - **styles.module.scss**
  - **scss/:** archivos de estilos generales de la aplicación como variables.
  - **services/:** archivos de definición de servicios (endpoints) para realizar las integraciones
  - **types/:** definiciones de tipos que serán usadas en toda la aplicación.
  - **utils/:** funciones utilitarias para la aplicación.

- … archivos de definiciones importantes de la aplicación como el componente inicial ***index.tsx***
- ... archivos de configuración de ***package.json, linters, gitignore, variables de ambiente, README, tsconfig, etc.***

GitFlow

Una vez definimos esta estructura estuvimos trabajando el equipo de frontend utilizando el siguiente GitFlow:

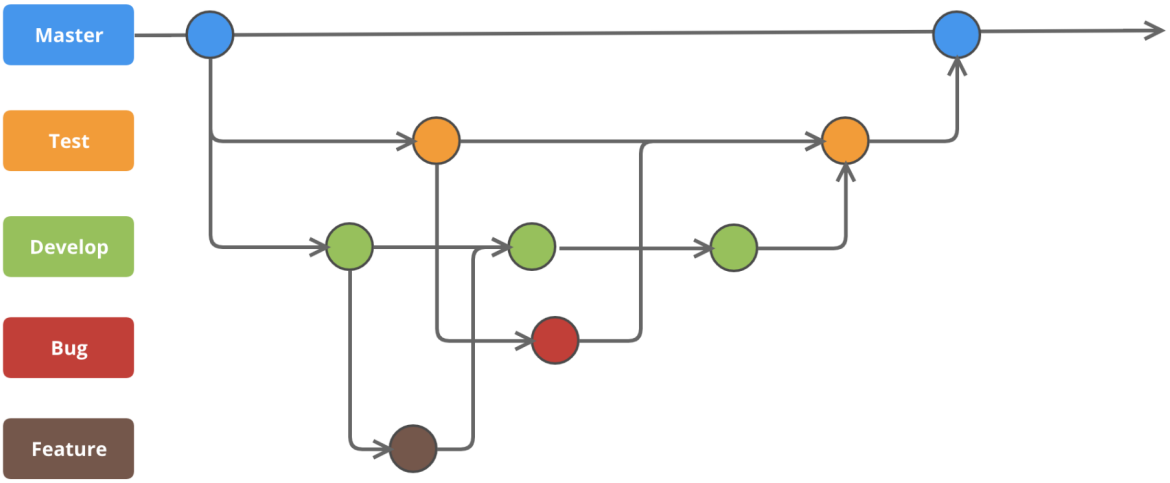

Figura 2. GitFlow Master/Test/Develop

El código inició a partir de una base alojada en la rama **master** que se lleva tanto a las ramas **test** como **develop**, la rama **develop** fue nuestro punto de partida en adelante para cualquier **feature** (funcionalidad) que desarrollamos. Adicionalmente, cuando se reportaba un bug en la aplicación se tomaba la rama **test** y a partir de allí se realizaba la respectiva corrección de este error, para finalmente volver a subirlo en una nueva versión de la misma rama test. Finalmente cuando se iba a desplegar una versión a producción se llevaba a la rama **master** de nuevo para que así el producto quedase desplegado y el usuario final pudiera acceder a esta nueva versión de la aplicación.

El equipo de infraestructura del cliente aprovisionó el CI/CD (Integración continua y despliegue continuo) en todas las ramas, lo que significa que cada vez que realizamos un merge (subir cambios a una rama) se realizaba el proceso de verificación de pruebas,

verificación de coverage de la aplicación y finalmente despliegue al ambiente correspondiente, para este proyecto teníamos 3 ambientes provisionados: **Develop, QA y Producción,** estos ambientes se generaban a partir de las ramas **Develop, Test y Master** respectivamente.

Durante el proyecto se tuvieron sesiones de desarrollo con horarios extraordinarios, las cuales fueron denominadas **War Room**, eran momentos críticos del proyecto donde debíamos llegar a una meta y para poder cumplirla destinamos un medio de comunicación inmediato como lo era Discord y así podríamos estar en contacto los diferentes equipos, entre frontend, backend y QA realizamos reuniones instantáneas para probar o llevar a cabo definiciones importantes, permitiendo que el equipo no se bloqueara por pequeños detalles y aumentando nuestra productividad.

Marvel

El equipo de diseño UI/UX utilizó a lo largo del proyecto la herramienta Marvel, una plataforma en línea que permite alojar mockups, que es una representación gráfica de cada una de las vistas de la aplicación, donde se detalla cómo espera que quede su diseño, color, tipografía, etc. “Al observar un mockup, se debe tener una buena idea de cómo se verá el producto final y una idea aproximada de cómo podría funcionar (incluso si las funciones aún no se han desarrollado).” **[9]** En el proceso de diseño el cliente estuvo interactuando activamente con esta plataforma, en cada una de las revisiones de diseño el **PO** llevaba sus comentarios y los anotaban directamente dentro de esta plataforma, esto ayudó a que hubiesen menos errores de comunicación ya que cuando un desarrollador comenzaba a desarrollar una funcionalidad no solamente podría ver el diseño esperado, sino que también podría ver los comentarios dejados por el cliente para esa funcionalidad en específico.

6. Resultados y análisis

El resultado de este proyecto fue el desarrollo de dos componentes principales de la aplicación, entre ellos se encuentra la autenticación, el cual fue desarrollado en su totalidad y se probó su funcionamiento tanto por el equipo de QA, como por el cliente, adicionalmente se realizaron pruebas de Ethical Hacking y pruebas manuales con algunos stakeholders, el resultado detallado de estas pruebas es de carácter confidencial, sin embargo se puede resumir que su resultado fue muy positivo para el cliente. A continuación detallaremos el diagrama de flujo para el componente de autenticación principal y todos los componentes o vistas asociadas a él

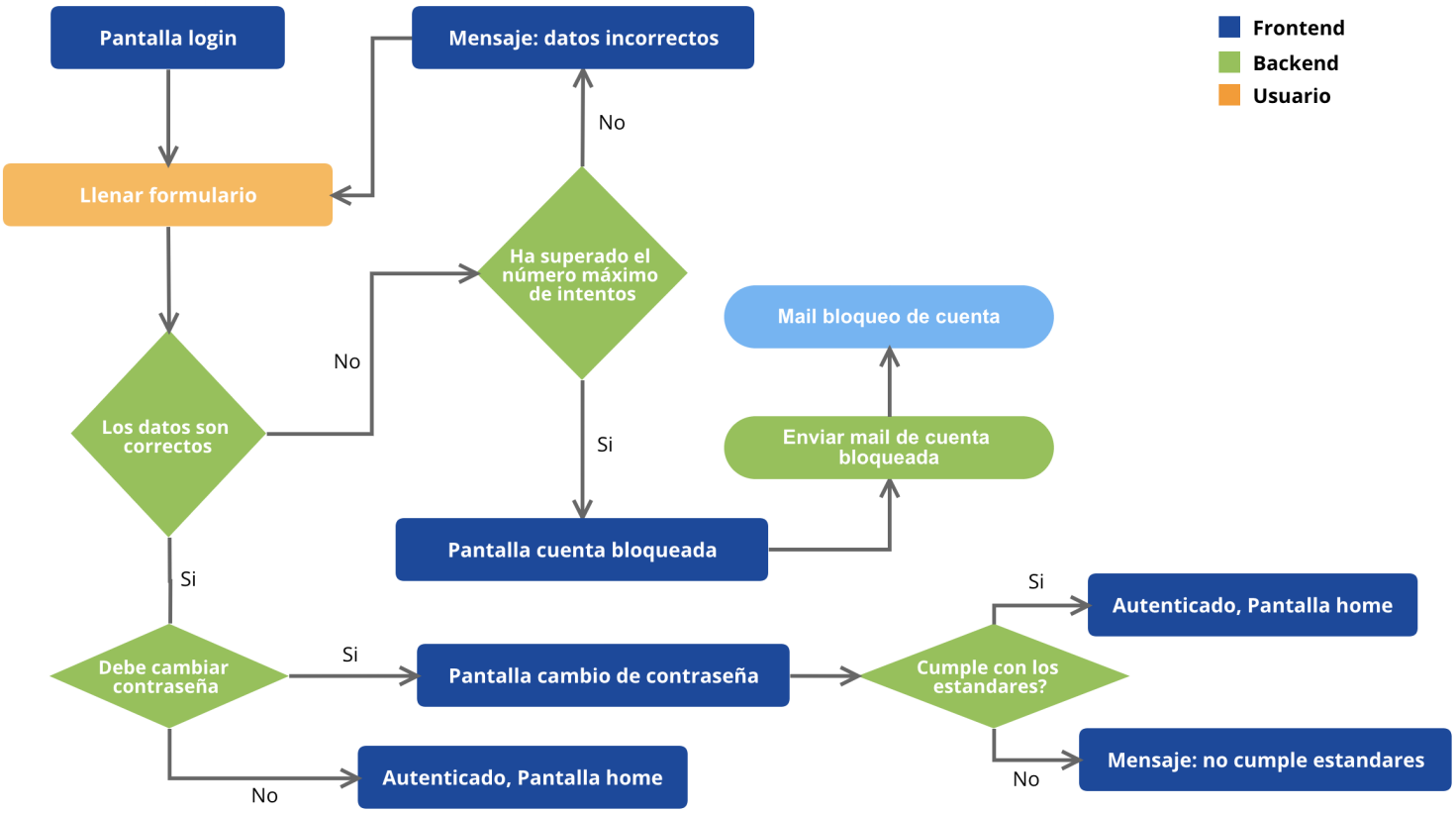

Figura 3. Flujo de Autenticación

En este flujo podemos apreciar que requería de una integración con los servicios de autenticación provistos por el backend. Se logró garantizar que el usuario pudiese conectarse con los datos de usuario y contraseña de su cuenta, que esta cuenta fuera bloqueada por seguridad en caso de haber más de 6 reintentos con contraseñas incorrectas y que el usuario restableciara su contraseña cada vez que

fuese necesario, en esta etapa del proyecto el usuario debía cambiar su contraseña si y sólo si era su primer ingreso a la aplicación o si cuenta había sido bloqueada anteriormente.

De igual forma tendremos los siguientes flujos para recuperar contraseña y cambio de contraseña:

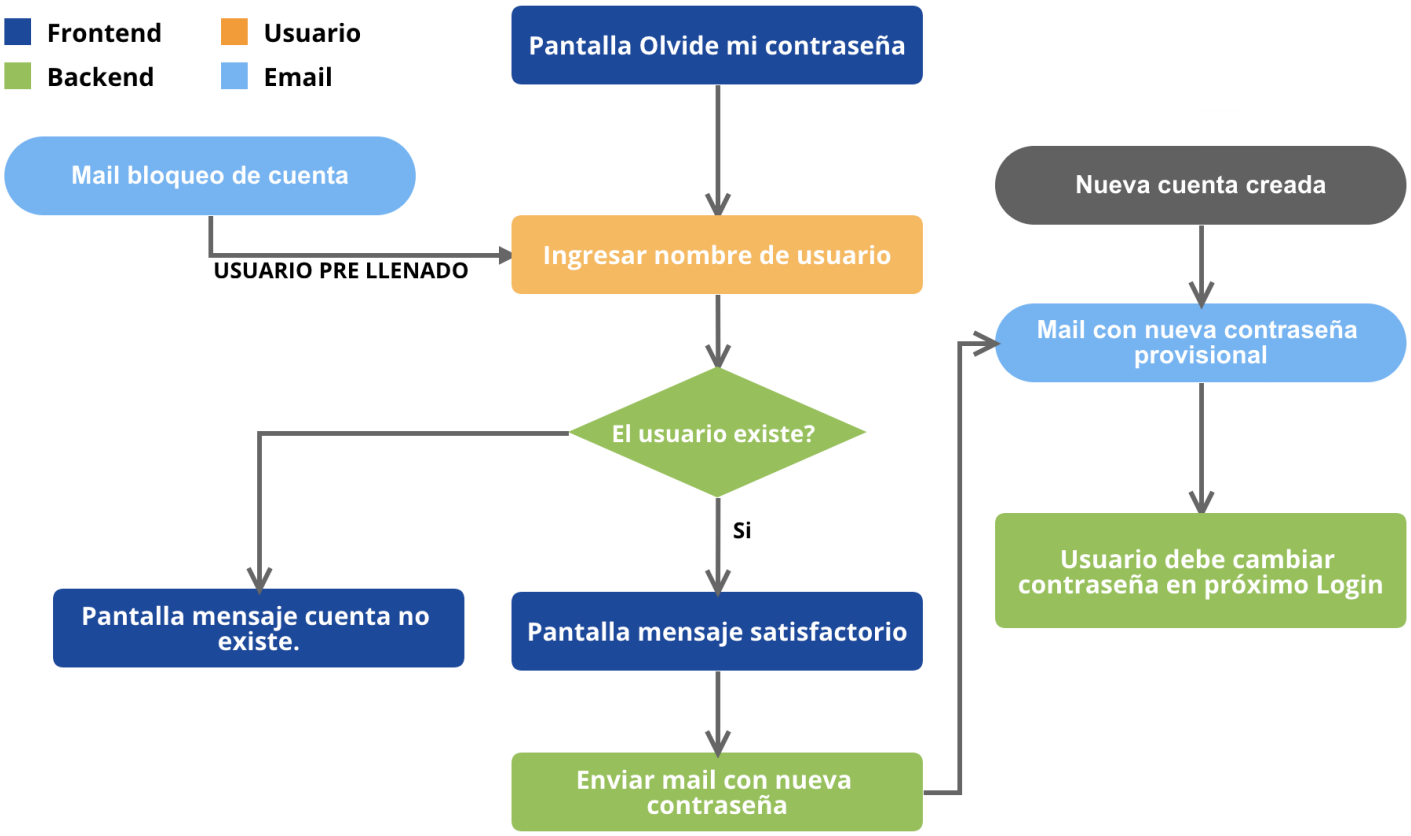

Figura 4. Flujo recuperar contraseña.

Con este flujo completado un usuario pudo iniciar sesión, recuperar su contraseña y cambiar su contraseña. Adicionalmente, debiamos garantizar la creación de un usuario dentro de la plataforma, para esto desarrollamos una vista de creación de usuario, en este proceso tuvimos en cuenta que tipos de roles existirían desde la aplicación, junto al cliente se pudieron definir los siguientes: Cliente, Agente y Superusuario. En este orden, los únicos usuarios que se pueden crear desde la aplicación son: Cliente y Agente, ya que por regla de negocio el usuario Superusuario sólo podría ser creado desde la base de datos.

A continuación detallaremos un mockup de la vista de creación de usuario, donde podremos ver los pasos que debe realizar un usuario de tipo Superusuario para crear un usuario Cliente o Agente:

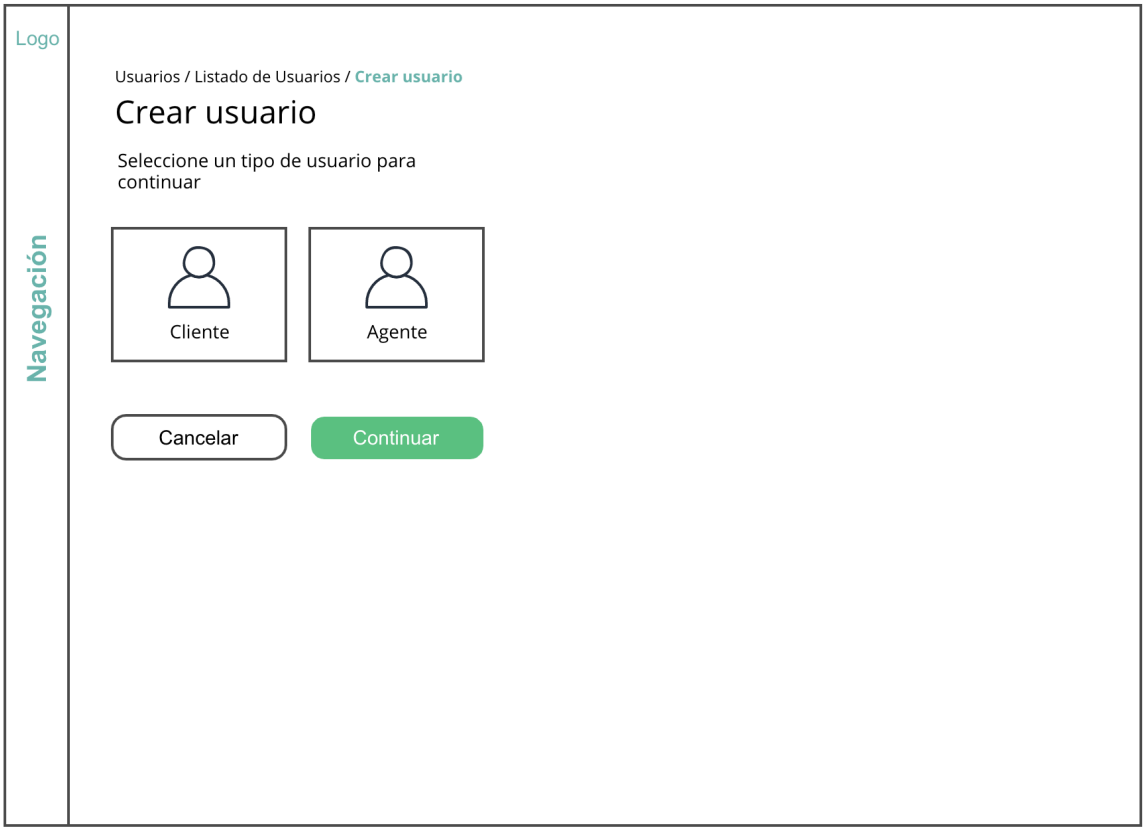

Figura 5. Mockup creación de usuario

En la pantalla principal el superusuario puede elegir que tipo de usuario va a crear, si es de tipo Cliente debería escoger una de las empresas para ser asociado a ella y si es de tipo agente continúa con el formulario y al final de este se le solicitará que vincule este agente con al menos una de las empresas que están registradas en la aplicación.

Para hacer este formulario se hizo uso del Context API de React donde guardabamos como estado el tipo de usuario a crearse y consecuentemente ir guardando la información que necesitaramos enviarle al Backend, uno de los grandes retos en este formulario fue la creación de un componente que permita seleccionar entre las empresas registradas en la aplicación, para desarrollar este componente el backend nos envía por medio de una petición la respuesta con un listado de empresas, este endpoint que nos provee el backend tiene a

su vez la funcionalidad de paginación, es por esto que nuestro selector de empresas debe ser interactivo, cuando un usuario presiona el botón de buscar se le despliega un listado inicial de empresas que corresponde a la primera página que nos entrega el endpoint de backend, una vez el usuario sigue haciendo scroll hacia abajo para buscar más empresas entonces el selector hace un nuevo request a backend para obtener la siguiente pagina y asi sucesivamente hasta completar el número de páginas disponibles en el backend. Así mismo, el selector debe tener la opción de poder buscar ingresando en el campo de texto el nombre de la empresa y debe arrojar el listado de empresas que correspondan al texto ingresado.

El flujo entonces de este módulo de creación de usuarios ha quedado así:

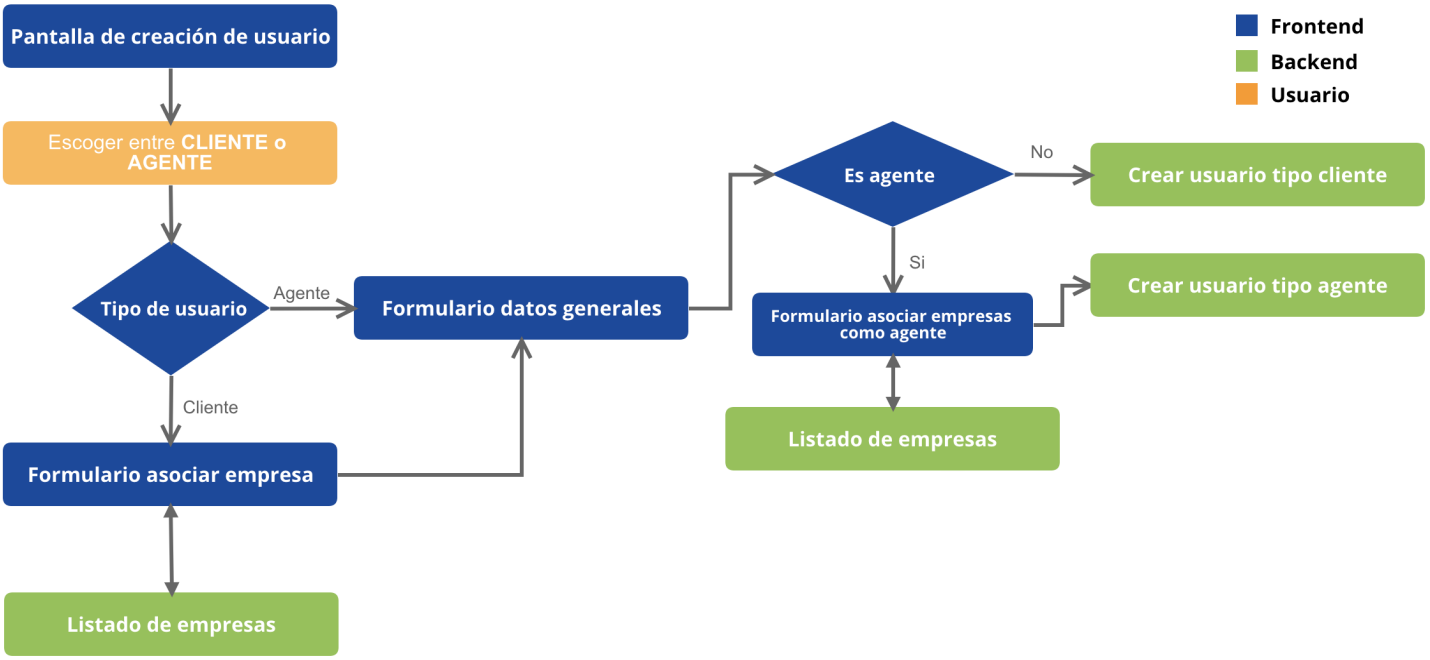

Figura 6. Flujo de creación de usuario

Teniendo en cuenta este flujo se desarrollan las vistas que permiten la creación y se verifica que la información se envíe correctamente al backend, siendo este el último responsable de realizar la creación en la base de datos y el proveedor de autenticación.

Finalmente se desarrolló el módulo de órdenes, comenzamos con el desarrollo de varios componentes necesarios para poder ver el detalle resumido una orden y luego combinamos esto con una pantalla donde se pudieran listar estas ordenes junto con un componente de paginación, finalmente añadimos el componente de búsqueda que a su vez incluía cada uno de los campos con los cuales se podría filtrar los resultados del listado de ordenes. A continuación mostramos el resultado final de este módulo con los componentes mencionados anteriormente:

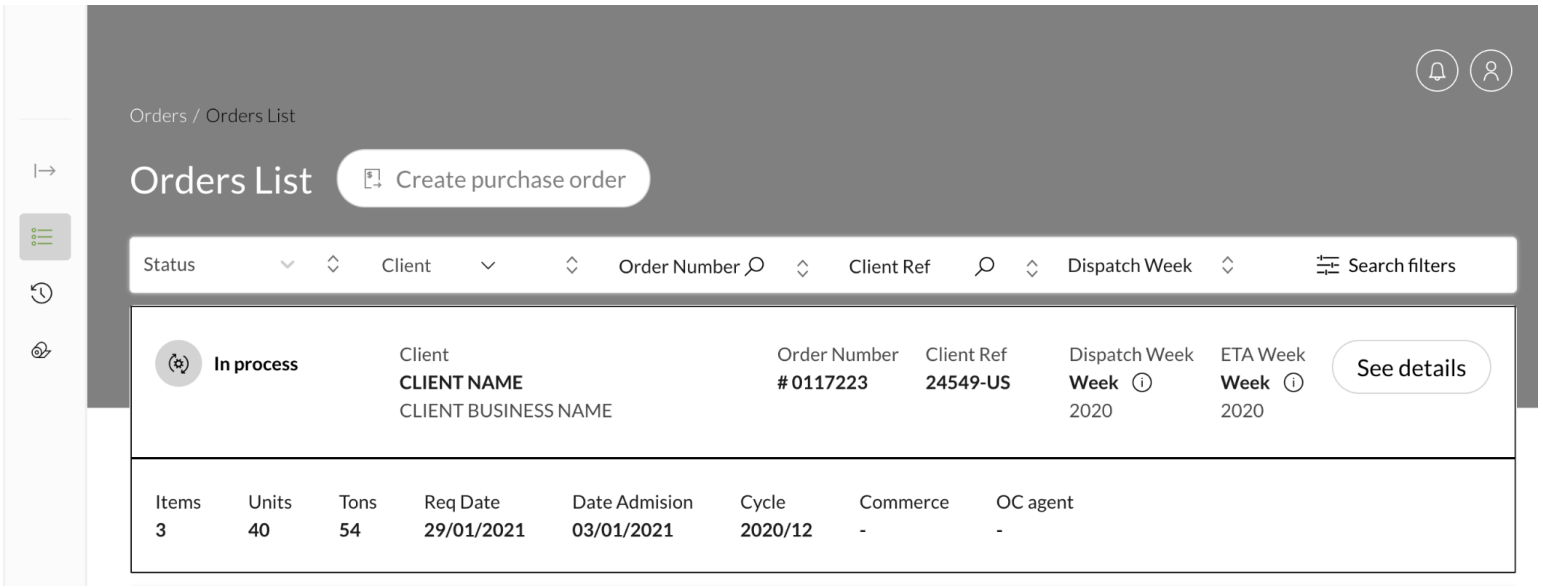

Figura 7. Modulo de Ordenes

Este módulo implicó gran cantidad de tiempo ya que como se puede apreciar en la figura posee bastante información de la orden que debió ser acomodada de forma que el usuario pudiera conocer todos los datos cómodamente legible. Adicionalmente el llamado a la acción que vemos en la parte derecha del componente debía realizar diferentes acciones dependiendo del estado en el que viniera la orden; para los casos donde la orden viniera en “Pendiente Validación” debería cambiar el texto del botón por “Validar” y al hacer clic sobre el deberia desplegar una nueva pantalla donde el usuario podría descargar la información de la orden en formato PDF y según esta información el usuario podría validar o rechazar esta orden.

En cuanto al trabajo realizado, el equipo de QA se encargó de probar todas las cards, garantizando que se cumpliera lo esperado en el diseño inicial, los deseos del cliente plasmados en las historias de usuario y el correcto funcionamiento de cada una de las funcionalidades. Este proceso iterativo se pudo ver reflejado en el siguiente gráfico:

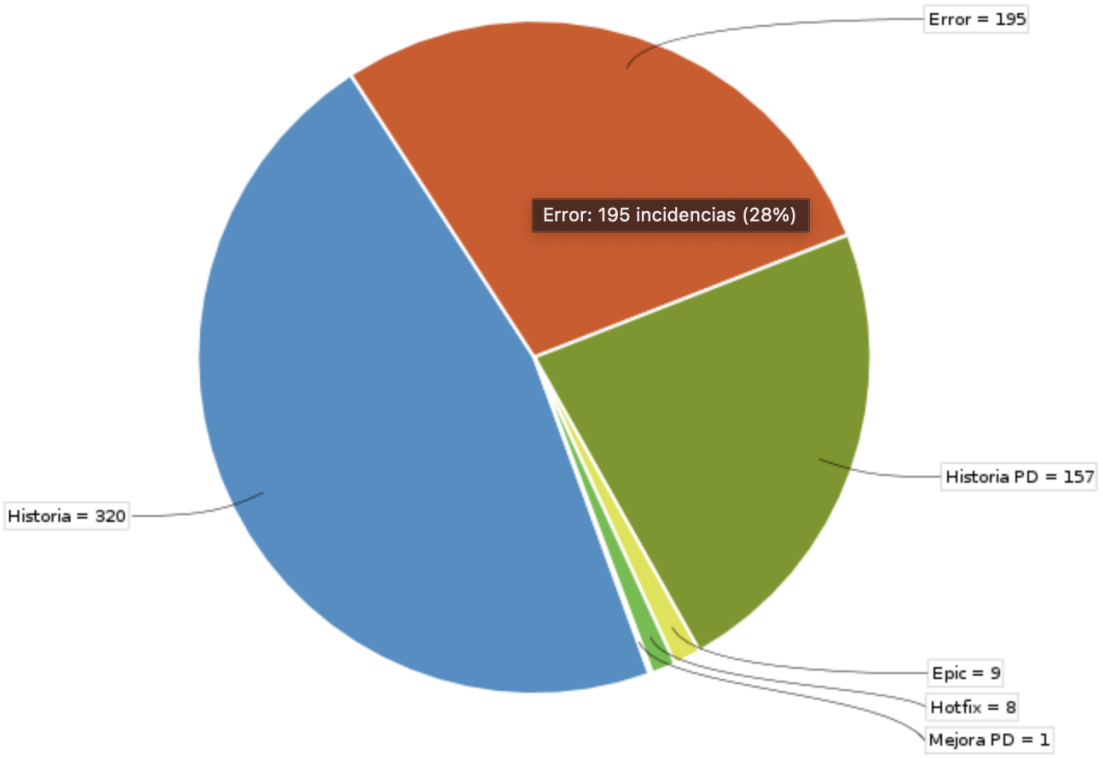

Figura 8. Gráfico incidencias por tipo

Aquí entonces vemos cómo del trabajo total realizado en las iteraciones que se completaron de este proyecto, un 28% estuvo dedicado a solucionar ese tipo de errores, un indicador que denota una buena labor por parte del equipo de QA analizando y detallando todos las posibles discordancias entre lo esperado y los resultados obtenidos en cada una de las historias de usuario.

7. Conclusiones

- En el módulo de autenticación se pudo garantizar que un usuario con nombre de usuario y contraseña válidos pudieran hacer uso de la plataforma, además que se brindaron las opciones de poder recuperar su contraseña y cambiar su contraseña. Así mismo, se garantizó la seguridad del aplicativo a partir del proceso de verificación por un tercero (ethical hacking) y además por estándares básicos de seguridad como el límite de intentos de login permitidos.

- El módulo de autenticación fue uno de los más satisfactorios para el proyecto, ya que no sólo se terminó en su totalidad, sino que también tuvo la oportunidad de ser probado por el equipo de QA, el cliente, un equipo de seguridad tercerizado que realizó pruebas de ethical hacking e incluso por algunos de los stakeholders del aplicativo.

- La metodología **SCRUM** presentó buenos resultados en general, el proyecto pudo ser entregado en los tiempos estimados y se cumplieron cada uno de los objetivos específicos pactados con el cliente en cada una de las iteraciones.

- El uso de librerías como React Hook Form facilitaron el desarrollo, brindando funciones y utilidades que permitieron completar las diferentes funcionalidades esperadas en un menor tiempo.

- Gracias al módulo de Órdenes los usuarios de la aplicación podrán listar las órdenes, además de poderlas filtrar por estado, cliente y demás campos relevantes de la orden. Además de que un usuario puede también entrar al detalle de una orden en específico para revisar no solo la información general sino todo el contenido de la misma, incluyendo los ítems que componen la orden y sus respectivos estados.

8. Referencias Bibliográficas

[1] Coleto Muñoz, M. (2018). ***Desarrollo de Aplicaciones Frontend con arquitectura Redux*** (Doctoral dissertation). Disponible en https://riunet.upv.es/handle/10251/106679

[2] Pérez, J. E. (2019). **Introduccion a JavaScript.** Disponible en http://190.57.147.202:90/xmlui/bitstream/handle/123456789/430/introduccion_javascript%20(1).pdf?sequence=1

[3] Jimenez, J. (2015). Descubre React. Recuperado en Octubre, 2, 2018. Disponible en http://www.academia.edu/download/60098162/Descubre_React__2da_Edicion_-_Javi_Jimenez-FREELIBROS20190723-17083-qn5tuy.pdf

[4] Loeliger, J., & McCullough, M. (2012). Version Control with Git: Powerful tools and techniques for collaborative software development. " O'Reilly Media, Inc.".. Disponible en https://books.google.es/books?hl=es&lr=&id=aM7-Oxo3qdQC

[5] de Sousa Antonio, C. (2015). Testing React Components. In Pro React (pp. 281-292). Apress, Berkeley, CA. Disponible en https://link.springer.com/chapter/10.1007/978-1-4842-1260-8_9

[6] Park, J., An, S., Youn, D., Kim, G., & Ryu, S. (2021). JEST: N+ 1-version Differential Testing of Both JavaScript Engines and Specification. arXiv preprint arXiv:2102.07498. Disponible en https://arxiv.org/abs/2102.07498

[7] Bahit, E. (2012). Scrum y eXtreme Programming para programadores. Disponible en: http://up-rid2.up.ac.pa:8080/xmlui/handle/123456789/2030

[8] Sverrisdottir, H. S., Ingason, H. T., & Jonasson, H. I. (2014). The role of the product owner in scrum-comparison between theory and practices. Procedia-Social and Behavioral Sciences, 119, 257-267. Disponible en https://www.sciencedirect.com/science/article/pii/S1877042814021211

[9] Salgado, C. (2015). Sketchs, mockups, wireframes y prototipos. Mosaic, (130), 3. Disponible en https://dialnet.unirioja.es/servlet/articulo?codigo=7472535