Revista Nexos Científicos *ISSN: 2773-7489*
Junio 2022 pp. 1-7 *Correo: editor@istvidanueva.edu.ec*
Volumen 1, Número 1 *URL: http://nexoscientificos.vidanueva.edu.ec/index.php/ojs/index*
Fecha de recepción: enero 2022 *Fecha de aceptación: marzo 2022*

Aplicación web para automatizar procesos de compra en la tienda Gramor Desing

Lizbeth Suarez[1]; Marta Salazar[2]; Luis Aguas[3]
[1]Instituto Superior Tecnológico Vida Nueva, Quito-Ecuador, lizbeth.suarez@istvidanueva.edu.ec,[2,3] Universidad Tecnológica Israel, Quito-Ecuador, msalazar@uisrael.edu.ec, aguaszoft@live.com

Resumen: La automatización de procesos es un tema que cada vez toma más importancia y hoy por hoy es una necesidad básica para cualquier persona que intente emprender. Esta automatización puede ser realizada mediante redes sociales como Facebook o servicios de mensajería como WhatsApp, sin embargo, estas opciones son poco personalizables. En este trabajo se presenta un sitio web realizado con las tecnologías de React y Node empleando la metodología de desarrollo Xtreme Computing y tomando un enfoque de interfaz minimalista para mejorar la experiencia de usuario. Los resultados fueron favorables para React y Node que demostraron ser tecnologías fáciles de aprender, pero al mismo tiempo efectivas y eficientes. El enfoque minimalista también obtuvo resultados favorables, que si bien es mejorable, funciona mejor que el enfoque de interfaces más coloridas que distraen al usuario.
Palabras clave: Tecnología, React, Node, HTTP, HTML

Web application to automatize buying process at Gramor design store

Abstract: Process automatization is a topic with a high importance current days, being it a basic necessity for people trying to start with a personal business or building a personal store. This automatization could be made through social networks such as Facebook or message services like WhatsApp, however, these options are not too much customizable. In this paper we introduce a web site built over React and Node using Xtreme Computing developing methodology, considering a minimalist approach for user interfaces trying to improve user experience. The results were favorable due to React and Node demonstrating to be easy to learn technologies but still effective and efficient. The minimalist approach got good results as well, even when it could be better and should improve in the future, it outperforms other approaches like the one of putting a lot of colors in the interface, distracting the user.
Keywords: Technology, React, Node, HTTP, HTML

1. INTRODUCCIÓN

Hoy en día el desarrollo de aplicaciones web está tomando más relevancia que nunca debido a la facilidad y rapidez con la que se crean y por su compatibilidad multiplataforma, pudiendo funcionar en cualquier sistema operativo y pantalla siempre y cuando se tenga acceso a un navegador web. Si bien la web y el desarrollo de páginas HTML ya llevan más de dos décadas existiendo, es en los últimos años que han tomado más relevancia debido a la interacción humano-máquina que se ha logrado con los nuevos avances tecnológicos, haciendo que cada vez los sistemas sean más complejos. En este sentido, resulta imperioso el uso de frameworks y librerías para facilitar el desarrollo de estas aplicaciones o sitios web, habiendo stacks de desarrollo con tecnologías tanto para el frontend, backend, como para el manejo de estados y bases de datos. React es una librería desarrollada y mantenida por Facebook cuyo objetivo es promover la reutilización del código lo más que se pueda, dando como resultado sitios web altamente eficientes, escalables, resilientes, y fáciles de modificar. Node.js es otra tecnología ampliamente utilizada en el campo del desarrollo de APIs y aplicaciones del lado del servidor, permitiendo ejecutar código JavaScript en un runtime de C+

Revista Nexos Científicos *ISSN: 2773-7489*
Junio 2022 pp. 1-7 *Correo: editor@istvidanueva.edu.ec*
Volumen 1, Número 1 *URL: http://nexoscientificos.vidanueva.edu.ec/index.php/ojs/index*
Fecha de recepción: enero 2022 *Fecha de aceptación: marzo 2022*

altamente eficiente con non blocking events. Para el manejo de estados se usará Redux, que está bien integrado con React y servirá para tener un mayor control sobre la interacción hombre-máquina y el flujo de información que este genera. En este sentido, las preguntas que se proponen son las siguientes:

- ¿Es el stack MERN (MySQL, Express, React, Node) eficiente en el contexto del desarrollo web?
- ¿Ayuda Redux a la gestión de datos en el sistema?
- ¿Mejora la experiencia el uso de una interfaz minimalista?
- ¿Existen dificultades en el desarrollo de un sitio web usando React y Node como tecnologías principales?

2. METODOLOGÍA

La metodología Extreme Programming XP también permite un desarrollo ágil y entregar funcionalidades del sistema parcial para mejorar en cada lanzamiento hasta obtener una versión estable que se pueda distribuir.

2.1.Planificación

Antes de planificar cualquier módulo se realizó un análisis de requisitos, el cuál mediante la estrategia de historia de usuario puede mostrar con mayor claridad las necesidades de este y así poder ofrecer un sistema que cumpla con lo esperado y no se dispare en costos.

2.2.Diseño

La arquitectura que se empleará será la arquitectura cliente servidor, la cual se muestra en la siguiente imagen (imagen 1. Arquitectura).

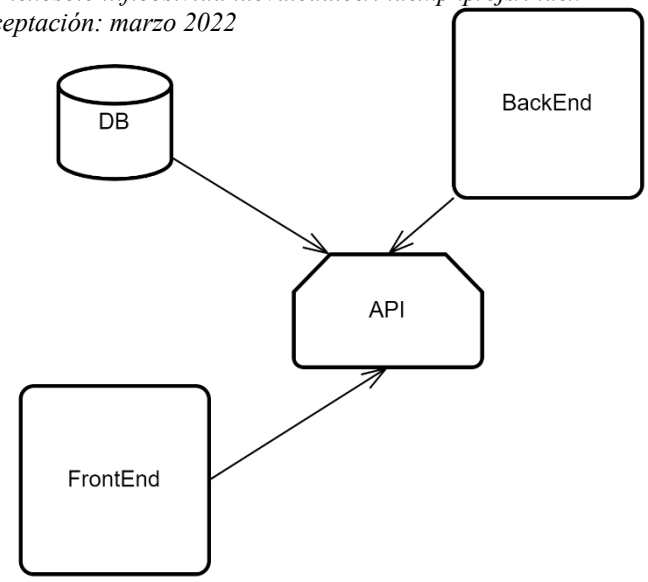
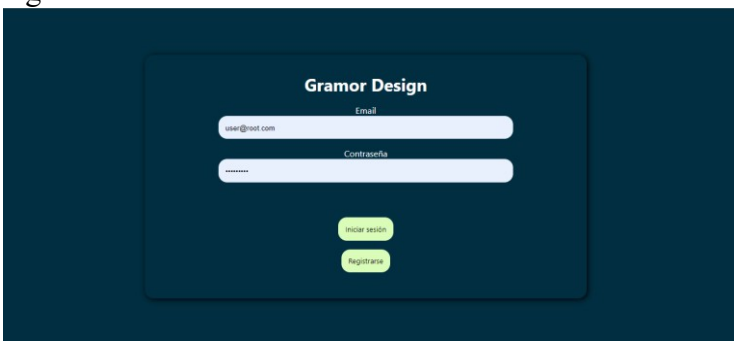


Imagen 1. Arquitectura

Este enfoque ayuda a mantener una aplicación modularizada y separar procesamiento de información de la interfaz de usuario final, además de que la API puede ser consumida por otras aplicaciones que pueden ser, por ejemplo, aplicaciones móviles.

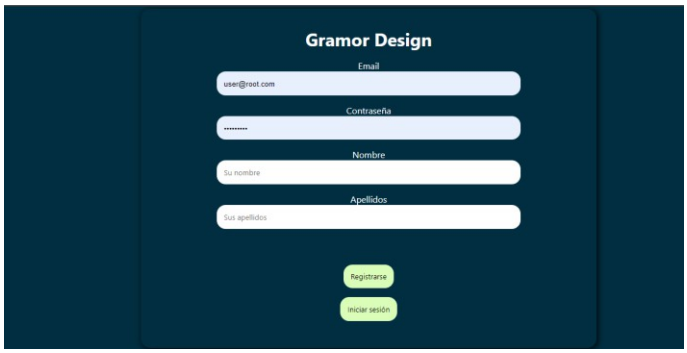
2.3.Desarrollo

Durante la etapa de desarrollo se consideró la siguiente interfaz de inicio de sesión.

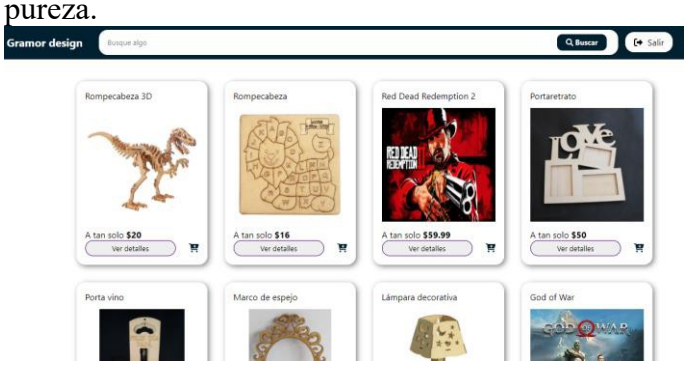


Esta interfaz fue pensada para ser lo más minimalista y limpia posible, focalizando la atención del usuario en los cuadros de inputs de email y contraseña, minimizando al máximo las distracciones. La siguiente imagen muestra la interfaz de registro, la cual es similar a la de login, pidiendo solo información de suma importancia como el nombre y apellido; email, y contraseña, requiriendo la menor cantidad de esfuerzo por parte del usuario.

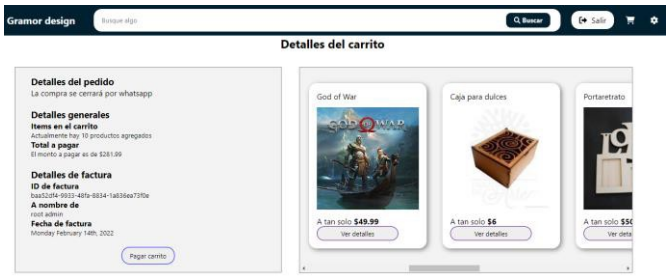
Revista Nexos Científicos *ISSN: 2773-7489*
Junio 2022 pp. 1-7 *Correo: editor@istvidanueva.edu.ec*
Volumen 1, Número 1 *URL: http://nexoscientificos.vidanueva.edu.ec/index.php/ojs/index*
Fecha de recepción: enero 2022 *Fecha de aceptación: marzo 2022*



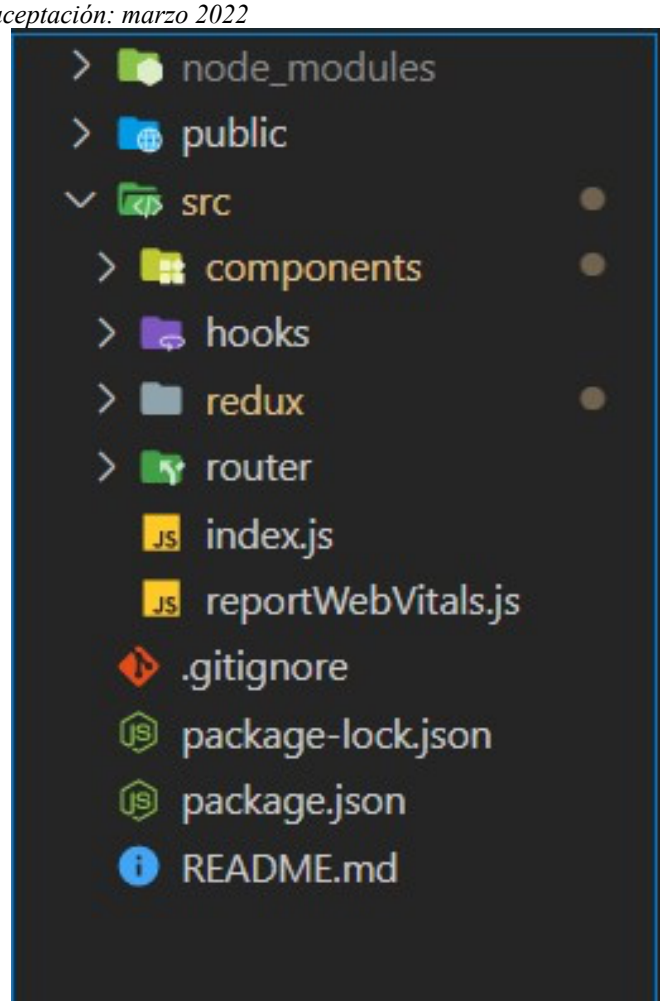
En la interfaz principal la prioridad estuvo en seguir al igual que el login con un diseño lo más minimalista posible, pero que sin embargo, reflejase pureza.



Esta página muestra los productos disponibles, haciendo todo lo posible por no contaminar visualmente la página. Si el usuario quiere más detalles del producto puede hacer click en el botón de "ver más", asimismo puede agregar el item al carrito con un solo click. La pantalla de detalles del pedido sigue el mismo principio, mostrando solo los datos esenciales de la factura.

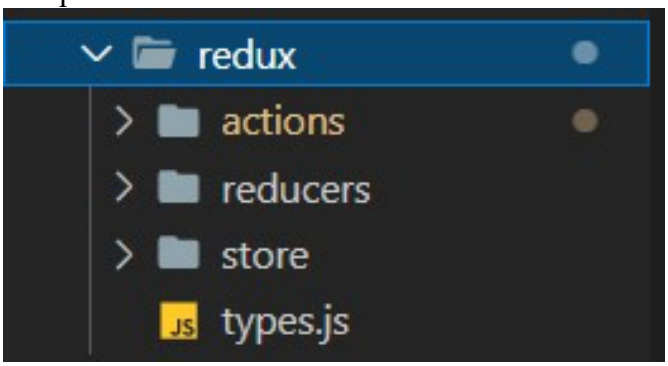


La arquitectura de la aplicación front-end cuenta con la siguiente distribución de carpetas.



De esta forma será más fácil escalar la aplicación ya sea vertical como horizontalmente.

Redux cuenta con cuatro módulos o componentes bien marcados que sirven para gestionar el flujo de información en toda la aplicación con un enfoque de computación reactiva.



En **actions** se encuentra todo lo relacionado a peticiones y cambios de estados de la aplicación, siendo este el único módulo capaz de modificar datos que entra en contacto directo con el usuario. Los **reducers** por otro lado, son los encargados de definir el cómo los **actions** van a afectar el estado de la aplicación. El **store** es el gran almacén de estados, aquí convergen los estados modificados por

Revista Nexos Científicos *ISSN: 2773-7489*
Junio 2022 pp. 1-7 *Correo: editor@istvidanueva.edu.ec*
Volumen 1, Número 1 *URL: http://nexoscientificos.vidanueva.edu.ec/index.php/ojs/index*
Fecha de recepción: enero 2022 *Fecha de aceptación: marzo 2022*

cada reducer para ser consumidos por los componentes de React mediante el uso de hooks. El archivo **types.js** es el encargado de darles un nombre a las acciones, para poder identificarlas en tiempo de ejecución.

El **store** luce de la siguiente manera, cumpliendo el papel de "base de datos" de los estados de la aplicación. Si un componente necesita algún estado, este tendrá que hace un query interno a este almacén de datos.

```
import thunk from "redux-thunk";
import {authReducer} from
"../reducers/authReducer";
import { carritoReducer } from
"../reducers/carritoReducer";
import { productReducer } from
"../reducers/productReducer";

const {combineReducers, createStore,
applyMiddleware, compose} =
require("redux");

// You should do this always since you
probably are
// gonna use several reducers in the
same app, and
// redux does not support many reducers
at the same
// time by default
const reducers = combineReducers({
  auth: authReducer,
  prod: productReducer,
  cart: carritoReducer,
});

// If you wana use several middlewares,
you should use next code
const composeEnhancers = (typeof window
!== 'undefined' &&
window.__REDUX_DEVTOOLS_EXTENSION_COMPOS
E__) || compose;

export const store = createStore(
  reducers,
  // this is necessary in order to show
redux status on your
  // browser. You don't need to do this
if you are not going
  // to analyze redux state on a
browser.
  //
  // here we're using thunk because you
are gonna be triggering
  // some asynchronous functions, so,
this middleware will help
  composeEnhancers(
    applyMiddleware( thunk )
  )
);
```

De la información extraída podemos realizar un arreglo de etiquetas links las cuales nos permiten determinar el total recolectado y recorrer cada uno de ellos y registrar en pantalla la información.

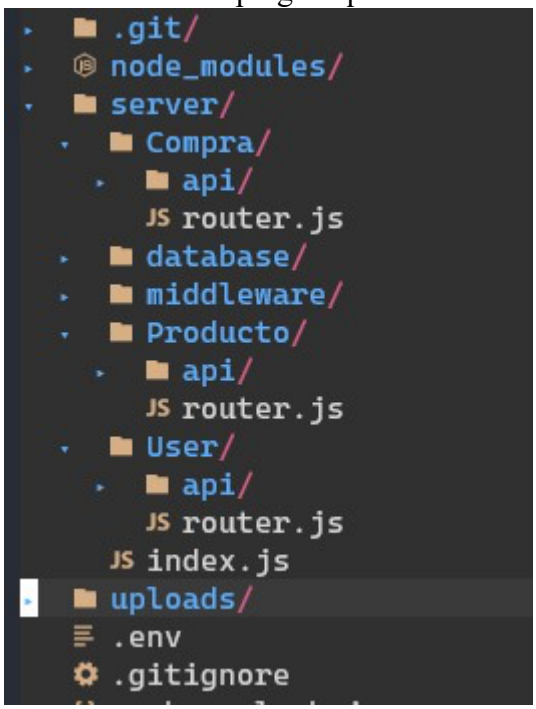
Las acciones son las encargadas de comunicarse con la API para que los **reducers** almacenen nuevos datos en forma de información, y como se observa, constan de dos partes, la primera, definida en el ejemplo como *startLoading* se encarga de pedir datos, y la segunda, definida como *setLoginData* se encarga de hacer un push al reducer correspondiente y actualizar el almacén de estados.

```
export const startLogin = (email,
pass)=>{
  return async(dispatch)=>{

    const res = await
fetch(`http://localhost:8500/user/login`
, {
      method: 'POST',
      body: new URLSearchParams({
        email, pass
      }),
      headers: {
        'Content-Type': 'application/x-
www-form-urlencoded; charset=UTF-8',
      }
    });


    const jsonRes = await res.json();

    if(!jsonRes.ok) {
      Swal.fire({
        title: "No se pudo iniciar
sesión",
```

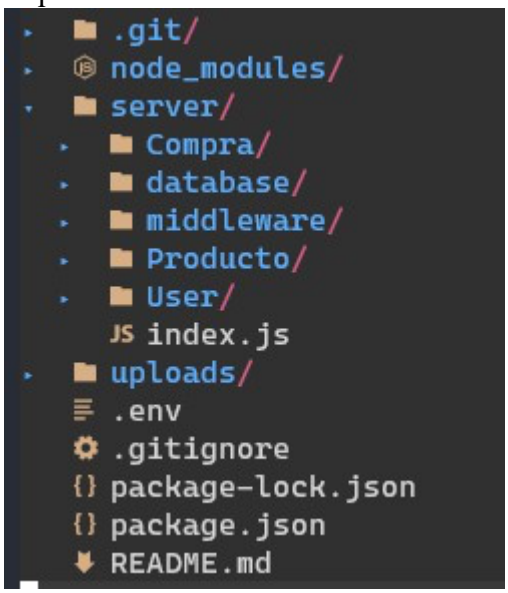
Revista Nexos Científicos
Junio 2022 pp. 1-7
Volumen 1, Número 1
Fecha de recepción: enero 2022
ISSN: 2773-7489
Correo: editor@istvidanueva.edu.ec
URL: http://nexoscientificos.vidanueva.edu.ec/index.php/ojs/index
Fecha de aceptación: marzo 2022

```
        text: jsonRes.mensaje,
        onClose: ()=>Swal.close()
      });
    }
    else{
      dispatch(
setLoginData(jsonRes.user,
jsonRes.token) );

      localStorage.setItem("nicostore-
token", jsonRes.token);
      localStorage.setItem("nicostore-
user", JSON.stringify(jsonRes.user));

      Swal.close();
      window.location.href =
"http://localhost:3000/home";
    }
  }
}

export const setLoginData = (userData,
token)=>({
  type: types.login,
  payload: {
    token,
    user: userData
  }
});
```

En este caso, se han dividido los servicios por módulos, con la intención de crear una arquitectura de microservicios en el futuro y optimizar los procesos y hacer el deploy más ligero. Cada módulo cuenta de momento con un *API*, que tiene las peticiones y sirve información al frontend; y un
archivo de enrutamiento, que se encarga de realizar que todos los módulos trabajen como uno solo y puedan comunicarse entre sí. Esta aplicación se desarrolló con ayuda de NeoVim dada su agilidad, mientras que React se realizó con ayuda de VS
code debido a los plugins que este editor ofrece.



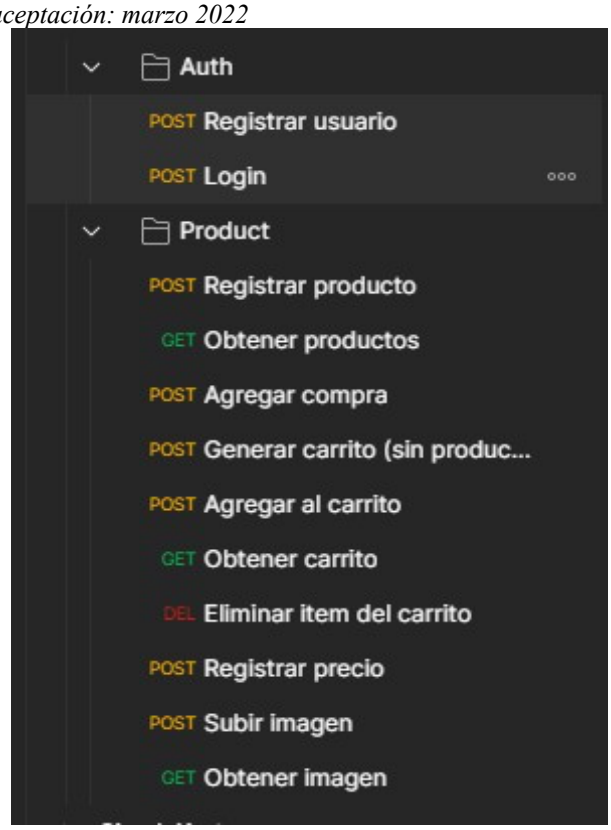
La arquitectura backend por otro lado cuenta con la arquitectura mostrada a continuación.



La base de datos cuenta con las siguientes tablas, permitiendo el uso de un carrito de compras. Fue desarrollada con el motor de InnoDB en MySQL y cuenta con triggers que valida el ingreso de datos en situaciones, como por ejemplo, cuando se intenta comprar un producto fuera de stock.

Revista Nexos Científicos *ISSN: 2773-7489*
Junio 2022 pp. 1-7 *Correo: editor@istvidanueva.edu.ec*
Volumen 1, Número 1 *URL: http://nexoscientificos.vidanueva.edu.ec/index.php/ojs/index*
Fecha de recepción: enero 2022 *Fecha de aceptación: marzo 2022*

```
mysql> show tables;
+-----------------------+
| Tables_in_nicole_store |
+-----------------------+
| Carrito               |
| CarritoCompras        |
| Compra                |
| Precio                |
| Producto              |
| User                  |
+-----------------------+
6 rows in set (0.05 sec)
```



2.4. Testing

El testing a la API se realizó usando la aplicación de Postman, que es ampliamente utilizada para el testeo y documentación de APIs, pudiendo segmentar las peticiones como se muestra a continuación y de esta forma poder desarrollar más cómodamente el front-end, dado que cada petición y respuesta se quedan guardadas.

En cuanto a las validaciones, todo el sistema hace uso de la librería validator de NPM, que se encarga de validar de forma eficiente diferentes campos como, por ejemplo, emails, contraseñas seguras, nombres, fechas, archivos, etc. El uso es tan simple como el que se da a continuación.

```
if(!validator.isEmail(req.body.email.trim())) valido =
"Email no valido";
```

La API valida internamente mediante el uso de json web tokens (JWT) que el usuario esté autorizado en realizar ciertas acciones, como por ejemplo, agregar productos, que es una tarea dedicada solo a usuarios con el rol ‘admin’.

Las peticiones testeadas en PostMan se muestran a continuación.

3. RESULTADOS Y DISCUSIÓN

EL uso de las librerías de React y Redux en conjunto con el runtime de Node.js y express han hecho posible la creación de un sitio web básico que sirve para realizar compras para la tienda **gramor design**. El sitio web resulta fácil de utilizar y gracias a la eficiencia provista por la librería de React, esta se siente bastante ligera al momento de usar. El proyecto cuenta con una arquitectura con base para microservicios y escalar tanto horizontal como verticalmente, pudiendo agregar nuevos módulos y funcionalidades, como, por ejemplo, incluir un sistema de reportes basado en inteligencia artificial con herramientas de machine learning y de esta forma saber qué productos poner en oferta o cuáles se espera vender más para realizar pedidos y aumentar las ganancias. Asimismo, se puede profundizar en lo ya existente y mejorarlo, por ejemplo, se puede incluir un motor de recomendación para modificar la forma en la que se muestran los productos (por ahora se muestran en orden alfabético) y así enganchar más al usuario. Se puede implementar la opción de favoritos o lista de deseados para de esta forma recordarle al usuario que tiene productos pendientes para comprar o notificarle cuando existen ofertas.

Existen varios sitios web similares realizados con otras tecnologías similares como Vue o Angular por parte del cliente, y Dino, Golang, Rust, entre otros lenguajes, por parte del servidor, sin embargo, en el caso del front-end, la comunidad de React es más activa y su curva de aprendizaje más rápida de acuerdo con lo investigado. En el lado del servidor si bien hay tecnologías más robustas como Rust o Golang, para este primer propósito Node cumplió con su objetivo, aunque quizás para futuras actualizaciones se pueda implementar algún módulo en otro lenguaje como Rust para evaluar posibilidades y diseñar módulos más seguros y rápidos, aprovechando la concurrencia ofrecida por Rust o Golang.

En cuanto a la experiencia de usuario, se le pidió a un conjunto de personas que probara el sistema y mencionasen si es agradable o no. La mayoría coincidió que el sistema es simple y fácil de usar, aunque sugirieron el uso de una paleta de colores distinta para las interfaces de login y registro de usuario.

4. CONCLUSIONES

- El stack MERN funciona bien para el desarrollo de aplicaciones web, al menos en un nivel básico, puesto que para sistemas más complejos se siente la necesidad del uso de tecnologías más robustas como Golang o Rust.
- Si bien Redux ayuda a la gestión de estados, para una aplicación no tan compleja como la propuesta se puede prescindir de su uso.
- La interfaz minimalista mejora la experiencia de usuario, aunque esta puede ser mejorada, sobre todo en las pantallas de login y registro.
- No se encontraron dificultades en el desarrollo de este sitio web usando las tecnologías de Node y React como las principales. Estas tecnologías se sintieron más ágiles frente Flutter y Rust.

REFERENCIAS

- Andrew, W. (11 de 06 de 2009). *informit.com.* Obtenido de https://www.informit.com/articles/article.aspx?p=1350956&seqNum=11
- B., G. (28 de 04 de 2020). *Hostinger*. Obtenido de Hostinger: https://www.hostinger.es/tutoriales/que-es-un-dominio-web
- Dordoigne, J. (2015). *Redes informáticas-Nociones fundamentales.* Ediciones Eni.
- DragonJAR. (05 de 2012). *dragonjar.org.* Obtenido de https://www.dragonjar.org/the-social-engineer-toolkit.xhtml
- Linux. (2002). *Trasparencias del Curso de IPv6.* GDU.
- Linux, K. (2018). *Penetration testing and ethical hacking linux distribution.*
- Lucia, P. G. (08 de 2018). *repository.uniminuto.edu.* Obtenido de https://repository.uniminuto.edu/bitstream/handle/10656/1091/TR_PedrazaGarzonCarmenLucia_2011.pdf?sequence=1&isAllowed=y
- Romero, J. G. (01 de 2019). *openaccess.uoc.edu.* Obtenido de http://openaccess.uoc.edu/webapps/o2/bitstream/10609/89045/6/joanenricgarciaromeroTFM0119memoria.pdf
- Vaca, M. (12 de 10 de 2012). *SET*. Obtenido de SET: https://www.hackplayers.com/2012/10/social-engineering-toolkit-set.html
- W, X., & H, J. (2014). Traffic Simulation Modeling and Analysis of BRT Based on Vissim. *Intelligent Computation Technology and Automation (ICICTA), 2014 7th International Conference* (págs. 879-882). IEEE.