Protocolo de entrega – Sprint 4

TEC Dev's Solutions

Integrantes:

- Laura Cárdenas Gómez – lauracardenasgomez@gmail.com
- Juan Pablo Maldonado – jpmaldonadop@misena.edu.co
- Marlon Mauricio Manrique Meza – mmanrique@misena.edu.co
- Jeremy Borgini Gutierrez Gómez – jbgutierrezg@unal.edu.co
- Edwin Rojas – edwinrojasf@gmail.com

Repositorio en GitHub

Luego de realizar la creación de las interfaces, se realizan los commits pertinentes en el repositorio de GitHub https://github.com/lauracardenasgomez/TICDevSolutions/tree/main

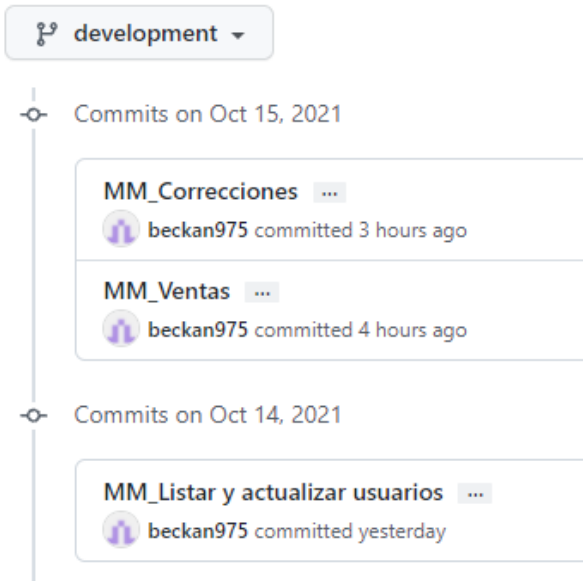

Imagen 1. Historial de commits del repositorio en GitHub

Todas las carpetas se están manejando en el Branch development para que, una vez ya completos los cambios pertinentes, se realice el merge al main Branch.

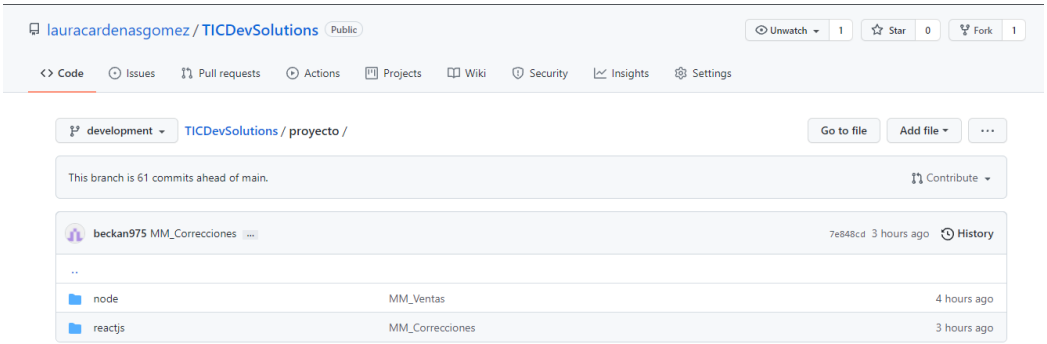

Imagen 2. Branch development

Seguimiento del tablero en Trello

Se realiza seguimiento diario o conforme a lo avanzado en el desarrollo del sprint del proyecto, el cual queda registrado en https://trello.com/b/CNELRm8E/sprint-4

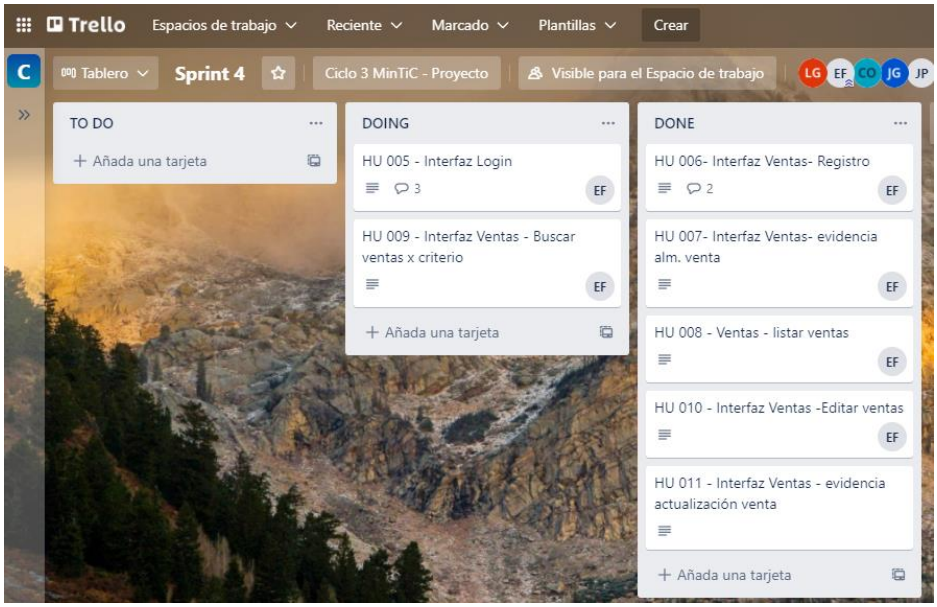

Imagen 3. Tablero Trello – Sprint 4

En el tablero se crean 7 historias de usuario para el Sprint 4 y se asignan los responsables para la realización de cada una de ellas. Los responsables de cada tarea diligencian su avance y se van terminando cada una de las historias de usario hasta tener todo el Sprint completo.

Descripción del proceso

En este sprint se buscaba que como rol vendedor o administrador se pudiera administrar la información de las ventas para gestionar la información del día a día. Para esto, se necesitaba validar las credenciales de usuario y poder ver las opciones para ingresar mediante Gmail.

Imagen 4. Autenticación

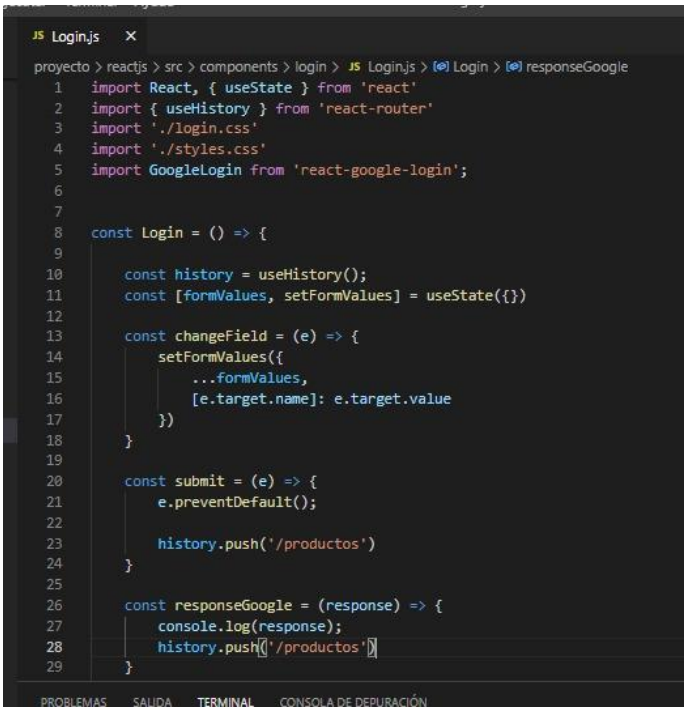

Imagen 5. Login autenticación

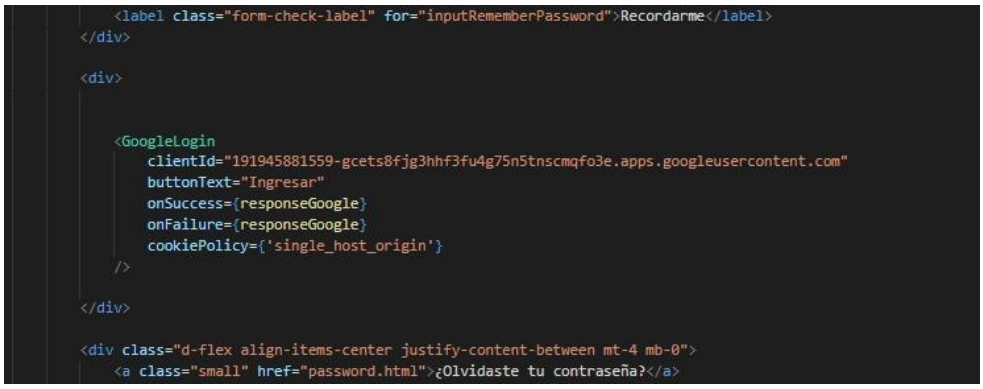

Imagen 6. Login autenticación

Una vez autenticadas las credenciales, se permite el ingreso al sistema de ventas al administrador o vendedor.

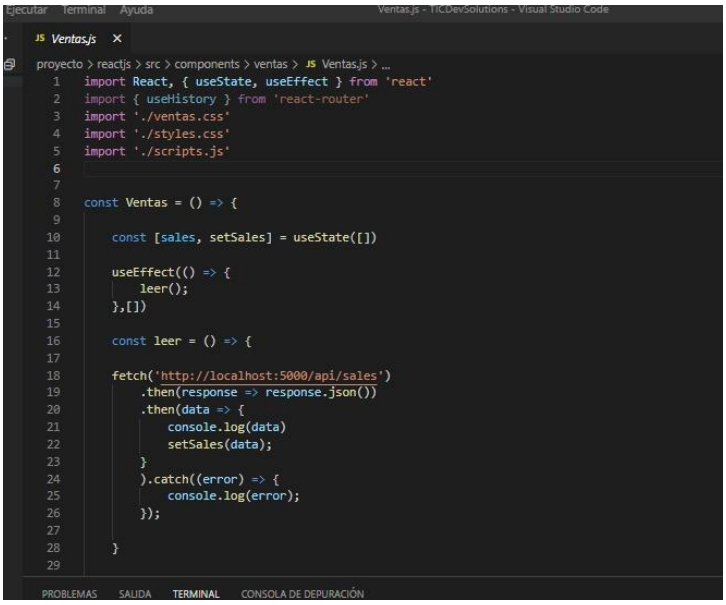

Imagen 4. Ventas

También se tiene el modelo de ventas y su controlador.

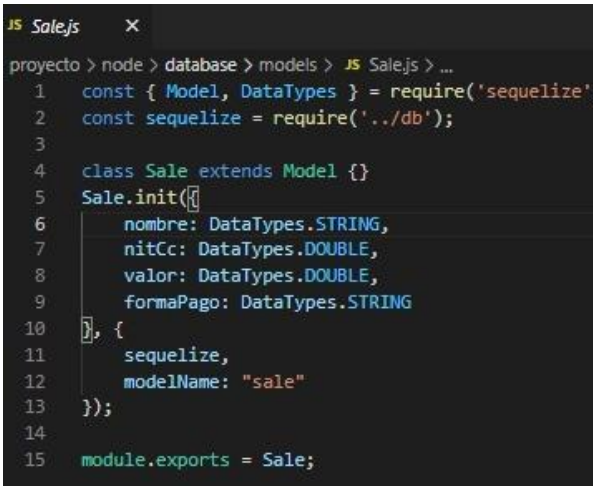

Imagen 5. Modelo ventas

Imagen 6. Ruta – controlador ventas

El rol administrador tendrá permisos para poder ingresar y actualizar los datos relacionados con un producto, de igual manera este puede ver la información del producto, realizar búsquedas mediante

el identificador de producto o su descripción, editar y actualizar información, al igual que poder ver que se almacenó correctamente en el sistema.

```
import React, { useState } from 'react'
import { useHistory } from 'react-router'
import './actualizarventa.css'
import './styles.css'
import './scripts.js'


const Actualizarventa = () => {


 const history = useHistory();
    const [formValues, setFormValues] = useState({})
    const [formTd, setFormId] = useState(0);

    const changeField = (e) => {
        setFormValues({
            ...formValues,
            [e.target.name]: e.target.value

        })
    }

    const submit = (e) => {
        e.preventDefault();
        console.log('formValues', formValues);

        fetch(`http://localhost:5000/api/sales/${formTd}`, {
            method: 'PATCH', // or 'PUT'
            body: JSON.stringify(formValues), // data can be `string` or {object}!
```

Imagen 5. Actualizar venta

```
import React, { useState } from 'react'
import { useHistory } from 'react-router'
import './buscarventa.css'
import './styles.css'
import './scripts.js'


const Buscarventa = () => {

    const [sales, setSales] = useState([])
    const [formId, setFormId] = useState(0);

    const submit = (e) => {
        e.preventDefault();
        fetch(`http://localhost:5000/api/sales/${formId}`)
            .then(response => response.json())
            .then(data => {
                console.log(data)
                setSales(data);

            }
            ).catch((error) => {
                console.log(error);
            });

        {
            (sales !== undefined && sales.length > 0) ?
            sales.map(item => {
                return (<tr>
```

Imagen 6. Buscar venta

```
import React, { useState } from 'react'
import { useHistory } from 'react-router'
import './registroventa.css'
import './styles.css'
import './scripts.js'


const Registroventa = () => {
    const history = useHistory();
    const [formValues, setFormValues] = useState({})

    const changeField = (e) => {
        setFormValues({
            ...formValues,
            [e.target.name]: e.target.value

        })
    }

    const submit = (e) => {
        e.preventDefault();
        console.log('formValues', formValues);

        fetch('http://localhost:5000/api/sales', {
            method: 'POST', // or 'PUT'
            body: JSON.stringify(formValues), // data can be `string` or {object}!
            headers: {
                'Content-Type': 'application/json'
            }
```

Imagen 7. Registro venta