

# Protocolo de entrega – Sprint 4

## TEC Dev's Solutions

---

### Integrantes:

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

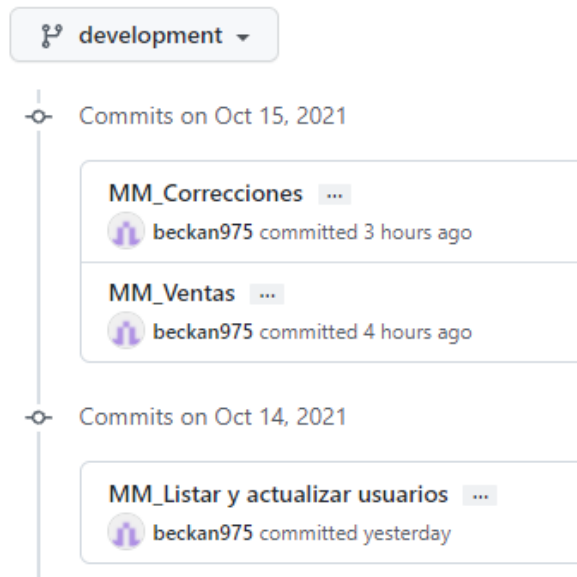
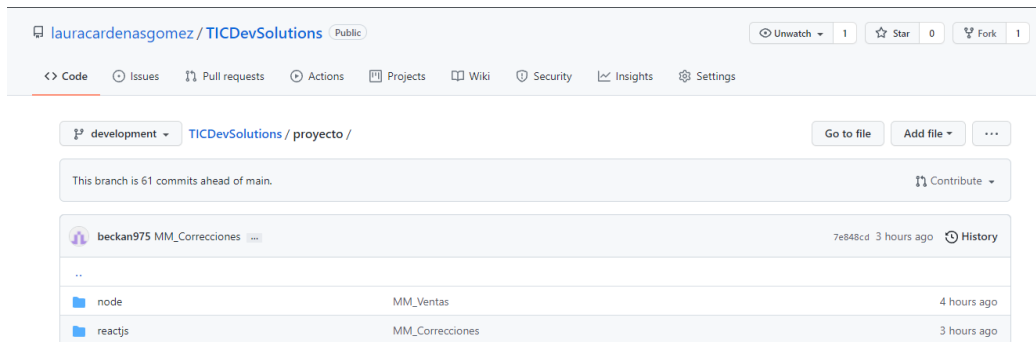


Imagen 1. Historial de commits del repositorio en GitHub

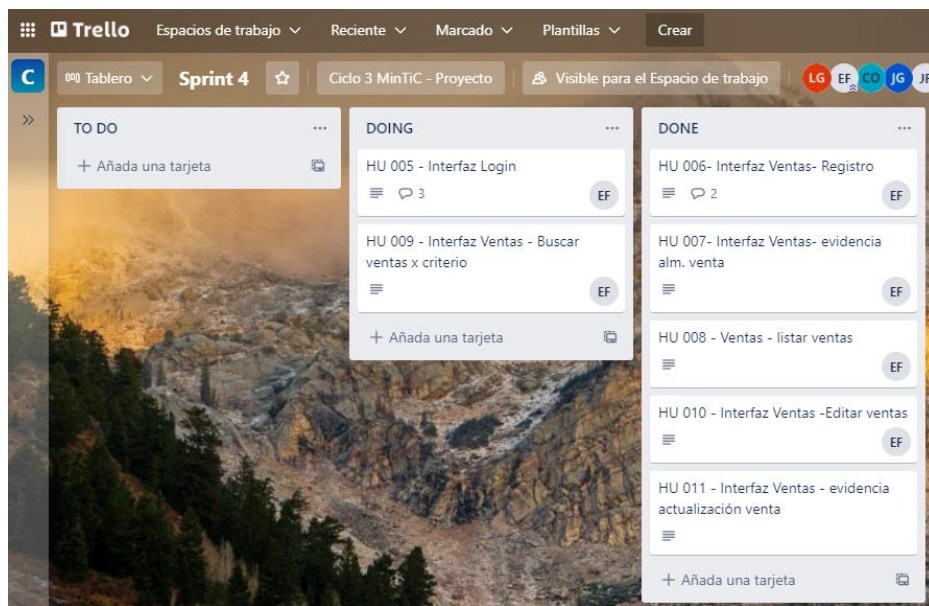
Todas las carpetas se están manejando en el Branch development para que, una vez ya completos los cambios pertinentes, se realice el merge al main Branch.



*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 usuario 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

```

JS Login.js x
proyecto > reactjs > src > components > login > JS Login.js > [0] Login > [0] responseGoogle
1  import React, { useState } from 'react'
2  import { useHistory } from 'react-router'
3  import './login.css'
4  import './styles.css'
5  import GoogleLogin from 'react-google-login';
6
7
8  const Login = () => {
9
10     const history = useHistory();
11     const [formValues, setFormValues] = useState({})
12
13     const changeField = (e) => {
14         setFormValues({
15             ...formValues,
16             [e.target.name]: e.target.value
17         })
18     }
19
20     const submit = (e) => {
21         e.preventDefault();
22         history.push('/productos')
23     }
24
25     const responseGoogle = (response) => {
26         console.log(response);
27         history.push('/productos')
28     }
29
30 }

```

Imagen 5. Login autenticación

```

<label class="form-check-label" for="inputRememberPassword">Recordarme</label>
</div>

<div>

  <GoogleLogin
    clientId="191945881559-gcets8fjg3hhf3fu4g75n5tnscmqfo3e.apps.googleusercontent.com"
    buttonText="Ingresar"
    onSuccess={responseGoogle}
    onFailure={responseGoogle}
    cookiePolicy={'single_host_origin'}
  />

</div>

<div class="d-flex align-items-center justify-content-between mt-4 mb-0">
  <a class="small" href="password.html">¿Olvidaste tu contraseña?</a>

```

Imagen 6. Login autenticación

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

```
JS Ventas.js x
proyecto > reactjs > src > components > ventas > JS Ventas.js > ...
1 import React, { useState, useEffect } from 'react'
2 import { useHistory } from 'react-router'
3 import './ventas.css'
4 import './styles.css'
5 import './scripts.js'
6
7
8 const Ventas = () => {
9
10   const [sales, setSales] = useState([])
11
12   useEffect(() => {
13     leer();
14   }, [])
15
16   const leer = () => {
17
18     fetch('http://localhost:5000/api/sales')
19       .then(response => response.json())
20       .then(data => {
21         console.log(data)
22         setSales(data);
23       })
24       .catch((error) => {
25         console.log(error);
26       });
27   }
28
29 }
```

Imagen 4. Ventas

También se tiene el modelo de ventas y su controlador.

```
JS Sale.js x
proyecto > node > database > models > JS Sale.js > ...
1 const { Model, DataTypes } = require('sequelize')
2 const sequelize = require('../db');
3
4 class Sale extends Model {}
5 Sale.init([
6   nombre: DataTypes.STRING,
7   nitCc: DataTypes.DOUBLE,
8   valor: DataTypes.DOUBLE,
9   formaPago: DataTypes.STRING
10 ], {
11   sequelize,
12   modelName: "sale"
13 });
14
15 module.exports = Sale;
```

Imagen 5. Modelo ventas

```
JS sales.js x
proyecto > node > routes > JS sales.js > ...
1 const express = require('express');
2 const router = express.Router();
3 const Sale = require('../database/models/Sale');
4 const Sequelize = require('sequelize');
5
6 // INDEX /api/sales
7 router.get('/', (req, res) => {
8   Sale.findAll().then(sales => {
9     res.json(sales);
10   })
11 })
12
13 // CREATE /api/sales
14 router.post('/', (req, res) => {
15   Sale.create({
16     nombre: req.body.nombre,
17     nitCc: req.body.nitCc,
18     valor: req.body.valor,
19     formaPago: req.body.formaPago
20   }).then(sale => {
21     res.json(sale);
22   })
23 });
24
25 // READ /api/sales/:id
26 router.get('/:id', (req, res) => {
27   Sale.findByPk(req.params.id).then(sale => {
28     res.json(sale);
29   })
30 })
```

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.

```
proyecto > reactjs > src > components > actualizarventa > JS Actualizarventa.js > Actualizarventa
1  import React, { useState } from 'react'
2  import { useHistory } from 'react-router'
3  import './actualizarventa.css'
4  import './styles.css'
5  import './scripts.js'
6
7
8  const Actualizarventa = () => {
9
10
11
12    const history = useHistory();
13    const [formValues, setFormValues] = useState({})
14    const [formId, setFormId] = useState(0);
15
16    const changeField = (e) => {
17      setFormValues({
18        ...formValues,
19        [e.target.name]: e.target.value
20      })
21    }
22
23    const submit = (e) => {
24      e.preventDefault();
25      console.log('formValues', formValues);
26
27      fetch('http://localhost:5000/api/sales/${formId}', {
28        method: 'PATCH', // or 'PUT'
29        body: JSON.stringify(formValues), // data can be 'string' or {object}!
```

Imagen 5. Actualizar venta

```
JS Buscarventa.js X
proyecto > reactjs > src > components > buscarventa > JS Buscarventa.js > Buscarventa
1  import React, { useState } from 'react'
2  import { useHistory } from 'react-router'
3  import './buscarventa.css'
4  import './styles.css'
5  import './scripts.js'
6
7
8  const Buscarventa = () => {
9
10
11    const [sales, setSales] = useState([])
12    const [formId, setFormId] = useState(0);
13
14    const submit = (e) => {
15      e.preventDefault();
16      fetch('http://localhost:5000/api/sales/${formId}')
17        .then(response => response.json())
18        .then(data => {
19          console.log(data)
20          setSales(data);
21        })
22        .catch(error => {
23          console.log(error);
24        });
25
26    {
27      (sales !== undefined && sales.length > 0) ?
28      sales.map(item => {
29        return <tr>
```

Imagen 6. Buscar venta

```
Ejecutar Terminal Ayuda Registroventa.js - TICDevSolutions - Visual Studio Code
JS Registroventa.js X
proyecto > reactjs > src > components > registroventa > JS Registroventa.js > Registroventa
1  import React, { useState } from 'react'
2  import { useHistory } from 'react-router'
3  import './registroventa.css'
4  import './styles.css'
5  import './scripts.js'
6
7
8  const Registroventa = () => {
9    const history = useHistory();
10    const [formValues, setFormValues] = useState({})
11
12    const changeField = (e) => {
13      setFormValues({
14        ...formValues,
15        [e.target.name]: e.target.value
16      })
17    }
18
19
20    const submit = (e) => {
21      e.preventDefault();
22      console.log('formValues', formValues);
23
24      fetch('http://localhost:5000/api/sales', {
25        method: 'POST', // or 'PUT'
26        body: JSON.stringify(formValues), // data can be 'string' or {object}!
27        headers: {
28          'Content-Type': 'application/json'
29        }
```

Imagen 7. Registro venta