

Protocolo de entrega – Sprint 3

TEC Dev's Solutions

Integrantes:

- Laura Cárdenas Gómez – auracardenasgomez@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/development>

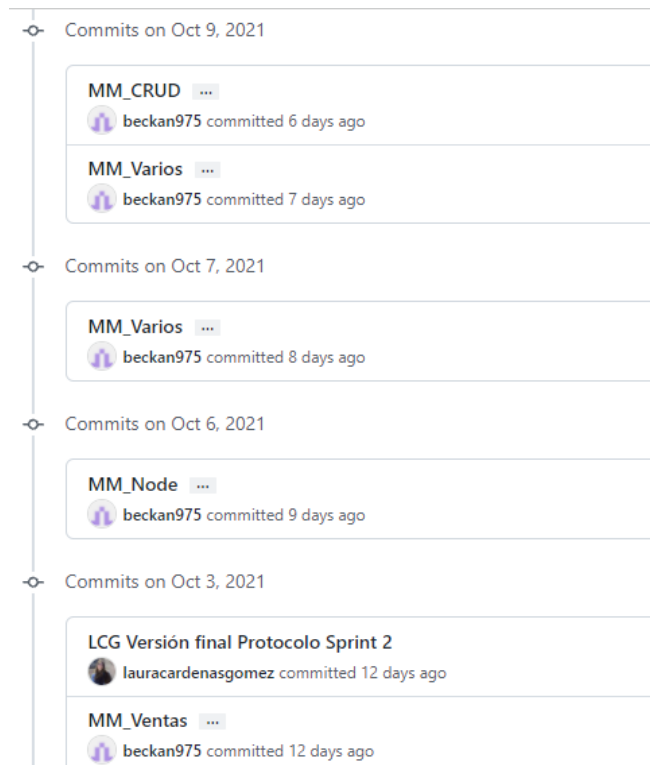


Imagen 1. Historial de commits del repositorio en GitHub

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.

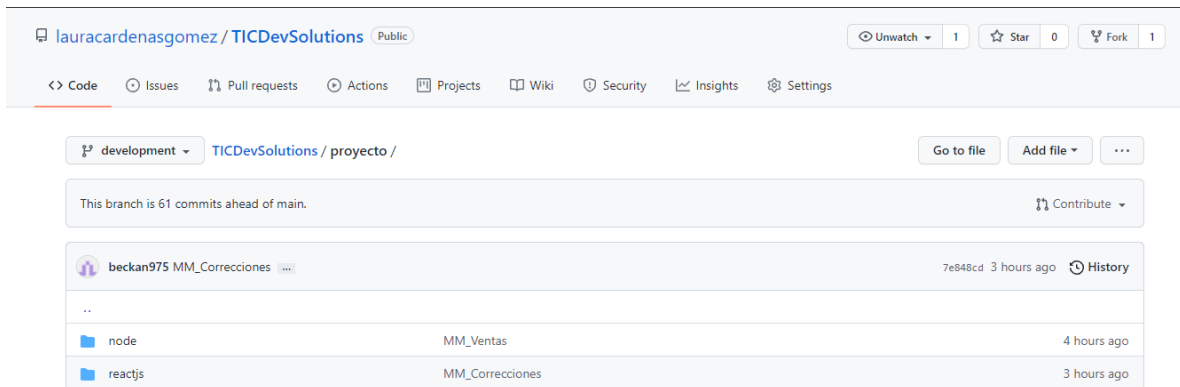


Imagen 2. Branch development

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/1yHUBTsN/sprint-3>

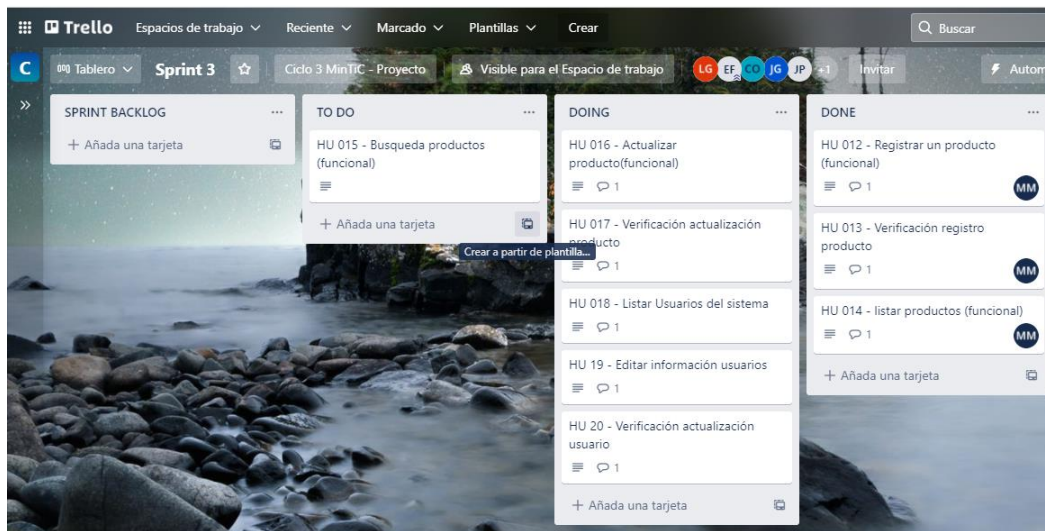


Imagen 3. Tablero Trello – Sprint 3

En el tablero se crean 9 historias de usuario para el Sprint 3 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

El objetivo de este sprint era: Como vendedor o administrador necesito administrar la información de las ventas para gestionar la información del día a día. Se crearon los modelos necesarios de producto y usuario con cada uno de los roles.

```

JS Product.js X
proyecto > node > database > models > JS Product.js > precioV
1  const { Model, DataTypes } = require('sequelize');
2  const sequelize = require('../db');
3
4  class Product extends Model {}
5  Product.init({
6    nombre: DataTypes.STRING,
7    //birthday: DataTypes.DATE
8    precioAd: DataTypes.DOUBLE,
9    precioV: DataTypes.DOUBLE
10 }, {
11   sequelize,
12   modelName: "product"
13 });
14
15 module.exports = Product;

```

Imagen 4. Modelo productos

```

JS User.js X
proyecto > node > database > models > JS User.js > password
1  const { Model, DataTypes } = require('sequelize');
2  const sequelize = require('../db');
3
4  class User extends Model {}
5  User.init({
6    nombre: DataTypes.STRING,
7    //birthday: DataTypes.DATE
8    email: DataTypes.STRING,
9    rol: DataTypes.STRING,
10   estado: DataTypes.STRING,
11   password: DataTypes.STRING
12 }, {
13   sequelize,
14   modelName: "user"
15 });
16
17 module.exports = User;

```

Imagen 5. Modelo usuarios

Esto permite que el administrador pueda 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.

```

JS products.js X
proyecto > node > routes > JS products.js > router.post('/') callback > precioV
1  const express = require('express');
2  const router = express.Router();
3  const Product = require('../database/models/Product');
4  const Sequelize = require('sequelize');
5
6  // INDEX /api/products
7  router.get('/', (req, res) => {
8    Product.findAll().then(products => {
9      res.json(products);
10    })
11  })
12
13  // CREATE /api/products
14  router.post('/', (req, res) => {
15    Product.create({
16      nombre: req.body.nombre,
17      precioAd: req.body.precioAd,
18      precioV: req.body.precioV
19    }).then(product => {
20      res.json(product);
21    })
22  });
23
24  // READ /api/products/:id
25  router.get('/:id', (req, res) => {
26    Product.findByPk(req.params.id).then(product => {
27      res.json(product);
28    })
29  });
30
31  // UPDATE /api/products/:id
32  router.patch('/:id', (req, res) => {
33    Product.update({
34      nombre: req.body.nombre,
35      precioAd: req.body.precioAd,
36      precioV: req.body.precioV
37    }, {
38      where: {
39        id: req.params.id
40      }
41    }).then(result => {

```

Imagen 6. CRUD productos - rutas

```

JS users.js X
proyecto > node > routes > JS users.js > ...
1  const express = require('express');
2  const router = express.Router();
3  const User = require('../database/models/User');
4
5  // INDEX /api/users
6  router.get('/', (req, res) => {
7    User.findAll().then(users => {
8      res.json(users);
9    })
10  })
11
12  // CREATE
13  router.post('/', (req, res) => {
14    User.create({
15      nombre: req.body.nombre,
16      email: req.body.email,
17      rol: req.body.rol,
18      estado: req.body.estado,
19      password: req.body.password
20    }).then(user => {
21      res.json(user);
22    })
23  });
24
25  // READ /api/users/:id
26  router.get('/:id', (req, res) => {
27    User.findByPk(req.params.id).then(user => {
28      res.json(user);
29    })
30  });
31
32  // UPDATE /api/users/:id
33  router.patch('/:id', (req, res) => {
34    User.update({
35      nombre: req.body.nombre,
36      email: req.body.email,
37      rol: req.body.rol,
38      estado: req.body.estado,
39      password: req.body.password
40    }, {
41      where: {

```

Imagen 7. CRUD usuarios - rutas

El proyecto desde del Sprint anterior se viene desarrollando en React y en este sprint se utilizó el entorno de la plataforma Node.js.

```
1  const express = require('express');
2  const app = express();
3  var cors = require('cors');
4  const sequelize = require('./database/db');
5
6
7  // Setting
8  const PORT = process.env.PORT || 5000;
9
10 // Middleware
11 // Para poder rellenar el req.body
12 app.use(express.json());
13 app.use(express.urlencoded({ extended: false }));
14 app.use(cors());
15
16 // Rutas
17 app.get('/', function (req, res) {
18   res.json("Hola Mundo");
19 });
20
21 app.use('/api/products', require('./routes/products'));
22 app.use('/api/users', require('./routes/users'));
23
24
25 // Arrancamos el servidor
26 app.listen(PORT, function () {
27   console.log('La app ha arrancado en http://localhost:${PORT}');
28
29   // Conectase a la base de datos
30   // Force true: DROP TABLES
31   sequelize.sync({ force: false }).then(() => {
32     console.log("Nos hemos conectado a la base de datos");
33   }).catch(error => {
34     console.log('Se ha producido un error', error);
35   })
36
37 });
```

Imagen 8. Servidor Node.js

```
1  import React, { useState } from 'react'
2  import { BrowserRouter, Switch, Route } from 'react-router-dom';
3  import logo from './logo.svg';
4  import './App.css';
5  import Login from './components/login/Login';
6  import Registro from './components/registro/Registro';
7  import Productos from './components/productos/Productos';
8  import Roles from './components/roles/Roles';
9  import Perfil from './components/perfil/Perfil';
10 import Registroproducto from './components/registroproducto/Registroproducto';
11 import Registroventa from './components/registroventa/Registroventa';
12 import Buscarproducto from './components/buscarproducto/buscarproducto';
13 import Buscarventa from './components/buscarventa/Buscarventa';
14 import Actualizarproducto from './components/actualizarproducto/Actualizarproducto';
15 import Actualizarventa from './components/actualizarventa/Actualizarventa';
16 import Ventas from './components/ventas/Ventas';
17
18
19
20
21 function App() {
22
23   return (
24     <div className="App">
25       <BrowserRouter>
26         <Switch>
27           <Route exact path="/" component={Login} />
28           <Route exact path="/login" component={Login} />
29           <Route exact path="/registro" component={Registro} />
30           <Route exact path="/productos" component={Productos} />
31           <Route exact path="/roles" component={Roles} />
32           <Route exact path="/perfil" component={Perfil} />
33           <Route exact path="/registroproducto" component={Registroproducto} />
34           <Route exact path="/registroventa" component={Registroventa} />
35           <Route exact path="/buscarproducto" component={Buscarproducto} />
36           <Route exact path="/buscarventa" component={Buscarventa} />
37           <Route exact path="/actualizarproducto" component={Actualizarproducto} />
38           <Route exact path="/actualizarventa" component={Actualizarventa} />
39           <Route exact path="/ventas" component={Ventas} />
40         </Switch>
41       </BrowserRouter>
```

Imagen 9. Rutas React