Tecnológico de Monterrey

Campus Querétaro

Contribuciones individuales de back-end

Marco Iván Flores Villanueva
A01276586

Materia: Construcción de software y toma de decisiones
Grupo: 501

Contribuciones individuales de back-end

Módulo de autenticación por roles de usuarios

El módulo de autenticación permite el acceso sólo a usuarios autenticados y con autorización de rol para ciertas vistas y rutas. Esto impide que los usuarios con rol de vendedor puedan realizar acciones de administración como la edición de datos protegidos.

```
// Crear plantilla
router.get('/crear', ensureAuthenticated, checkRole( roles: [2]), plantillaController.vistaCrearPlantilla);

// Ver plantillas guardadas
router.get('/guardadas', ensureAuthenticated, plantillaController.vistaPlantillasGuardadas);

// Detalle de una plantilla específica
router.get('/:id_live/detalle', ensureAuthenticated, checkRole( roles: [2]), plantillaController.vistaDetallePlantilla);

// Guardar plantilla
router.post('/guardar', ensureAuthenticated, checkRole( roles: [2]), plantillaController.guardarPlantilla);

// RUTAS PARA EDICION Y BORRADO DE PLANTILLAS
router.put('/:id', ensureAuthenticated, checkRole( roles: [2]), plantillaController.actualizarPlantilla);
router.delete('/:id', ensureAuthenticated, checkRole( roles: [2]), plantillaController.eliminarPlantilla);
```

La lógica de este módulo se encuentra descrita en el archivo *auth.middleware.js*, haciendo que sea necesario importar este middleware a todos los demás archivos de rutas.

```
exports.ensureAuthenticated = (req, res, next) => {
    if (req.isAuthenticated()) return next();
    res.redirect('/login');
};

exports.checkRole = (roles) => {
    return (req, res, next) => {
        if (roles.includes(req.user?.id_rol)) return next();
        res.status(403).send('No tienes permisos');
    };
};

exports.isAdmin = (req, res, next) => {
    if (req.isAuthenticated() && req.user?.id_rol === 1) {
        return next();
    }
    return res.status(403).json({ mensaje: 'Acceso denegado: sólo administradores' });
};
```

```
const { ensureAuthenticated, checkRole } = require('../middleware/auth.middleware');
```

Módulo de inicio de sesión y registro de usuarios

La lógica de este módulo se encuentra repartida en los archivos de rutas y controladores del módulo *usuario*. La lógica de los login funciona tanto para vendedores como para administradores, pues ambos tipos de usuarios, aunque posean roles diferentes, tienen en común que poseen un correo y una contraseña definida por ellos que pueden utilizar en la misma vista de login con el mismo back-end. Sin embargo, sólo es posible registrar vendedores, pues ellos son los principales usuarios del sistema, y después de adquirir una invitación mediante el *módulo de invitaciones*, son capaces de crear su usuario en el sistema

para posteriormente comenzar la creación de sus plantillas de productos mediante el *módulo de plantillas*.

```
exports.vistaLogin = (req, res) => {
    res.render('login', { error: null });
};

exports.vistaRegister = (req, res) => {
    res.render('register', { error: null });
};

exports.logout = (req, res) => {
    req.logout(() => {
        res.redirect('/');
    });
};

exports.registrarUsuario = async (req, res) => {...};
```

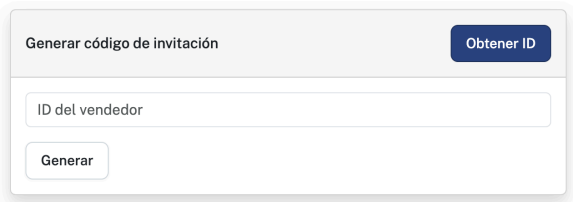
```
const express = require('express');
const router = express.Router();
const passport = require('passport');
const usuarioController = require('../controllers/usuario.controllers');

// Vistas
router.get('/login', usuarioController.vistaLogin);
router.get('/register', usuarioController.vistaRegister);
router.get('/logout', usuarioController.logout);

// Procesar login
// Procesar login con redirección dinámica según el rol
router.post('/login', (req, res, next) => {  ± Ivan
    passport.authenticate( strategy: 'local',  options: (err, user, info) => {
        if (err) return next(err);
        if (!user) return res.redirect('/login');

        req.logIn(user, (err) => {
            if (err) return next(err);
            // Redirección según el rol
            if (user.id_rol === 1) {
                return res.redirect('/dashboard');
            } else if (user.id_rol === 2) {
                return res.redirect('/dashboard-vendedor');
            } else {
                return res.status(403).send('Rol no permitido');
            }
        });
    })(req, res, next);
});


// Procesar registro
router.post('/register', usuarioController.registrarUsuario);

module.exports = router;
```

Módulo de invitaciones

Este módulo permite la invitación de vendedores mediante su user ID único de facebook, pues de esta manera es posible monetizar el sistema y "vender" el acceso al mismo a aquellos usuarios interesados. Una vez que el administrador llegue a un trato con un vendedor interesado, podrá utilizar el módulo de invitaciones para generar un hash único que el vendedor tendrá que utilizar para registrarse. Esto, además de monetizar el acceso al sistema, incrementa la seguridad, pues los registros ahora son controlados y restringidos a ciertos usuarios de confianza. La lógica de este módulo se encuentra en los archivos de rutas y controladores *admin*.

```
const db = require('../db');
const crypto = require('crypto');

exports.generarInvitacion = async (req, res) => {
    const { id_vendedor } = req.body;

    if (!id_vendedor) {
        return res.status(400).json({ mensaje: 'Falta el ID del vendedor', bodyRecibido: req.body });
    }

    try {
        const conn = await db.getConnection();

        const existente = await conn.query( sql: 'SELECT * FROM invitacion WHERE id_vendedor = ?',  values: [id_vendedor]);
        if (existente.length > 0) {
            conn.release();
            return res.status(409).json({
                mensaje: 'Ya existe un código para este vendedor',
                codigo: existente[0].codigo_hash
            });
        }

        const codigo = crypto.randomBytes( size: 8).toString( encoding: 'hex');

        await conn.query(
            sql: 'INSERT INTO invitacion (id_vendedor, codigo, fecha_creacion) VALUES (?, ?, NOW())',
            values: [id_vendedor, codigo]
        );

        conn.release();
        return res.status(201).json({ mensaje: 'Código generado', codigo });
    } catch (error) {
        console.error('Error al generar invitación:', error);
        return res.status(500).json({ mensaje: 'Error interno del servidor' });
    }
};
```

```
const express = require('express');
const router = express.Router();
const adminController = require('../controllers/admin.controllers');
const { isAdmin } = require('../middleware/auth.middleware');

router.post('/generar-invitacion', isAdmin, adminController.generarInvitacion);

module.exports = router;
```



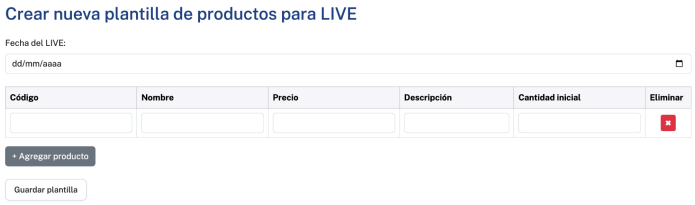
Módulo de plantillas e importación

Después de conocer la especificación del socio formador que dictaba que cada vendedor podría cambiar su catálogo en cada LIVE, descartamos la posibilidad de tener una tabla en nuestra base de datos que almacenara los productos de los vendedores como si fuera un catálogo, pues como ahora se sabía que los vendedores podían cambiar su catálogo sin previo aviso y modificar sus códigos de productos, propuse crear un sistema de plantillas que funcionaran como catálogos dinámicos que se tendrían que seleccionar para cada importación de comentarios CSV. De esta manera, los vendedores con sesión iniciada podrían crear sus propias plantillas con los productos que mostrarán o mostraron en diferentes LIVEs para que después los administradores, al momento de importar los mismos, sólo tuvieran que seleccionar la plantilla correspondiente sin temor a que ciertos productos mencionados en el LIVE ya no existieran en el catálogo de los vendedores.

Módulo de plantillas:

```
const express = require('express');
const router = express.Router();
const { ensureAuthenticated, checkRole } = require('../middleware/auth.middleware');
const plantillaController = require('../controllers/plantillas.controllers');

// Crear plantilla
router.get('/crear', ensureAuthenticated, checkRole( roles: [2]), plantillaController.vistaCrearPlantilla);

// Ver plantillas guardadas
router.get('/guardadas', ensureAuthenticated, plantillaController.vistaPlantillasGuardadas);

// Detalle de una plantilla especifica
router.get('/:id_live/detalle', ensureAuthenticated, checkRole( roles: [2]), plantillaController.vistaDetallePlantilla);

// Guardar plantilla
router.post('/guardar', ensureAuthenticated, checkRole( roles: [2]), plantillaController.guardarPlantilla);

// RUTAS PARA EDICION Y BORRADO DE PLANTILLAS
router.put('/:id', ensureAuthenticated, checkRole( roles: [2]), plantillaController.actualizarPlantilla);
router.delete('/:id', ensureAuthenticated, checkRole( roles: [2]), plantillaController.eliminarPlantilla);

module.exports = router;
```

```
const pool = require('../db');

// GET /plantillas/crear
exports.vistaCrearPlantilla = (req, res) => {
    res.render('plantillas/crear', { user: req.user });
};

exports.vistaPlantillasGuardadas = async (req, res) => { ... };

// GET /plantillas/:id_live/detalle
exports.vistaDetallePlantilla = async (req, res) => { ... };

exports.actualizarPlantilla = async (req, res) => { ... };


// POST /plantillas/guardar
exports.guardarPlantilla = async (req, res) => { ... };

// DELETE /plantillas/:id
exports.eliminarPlantilla = async (req, res) => { ... };
```



Módulo de importación:

```
const express = require('express');
const router = express.Router();
const multer = require('multer');
const path = require('path');
const importarController = require('../controllers/importar.controllers');
const { ensureAuthenticated, checkRole } = require('../middleware/auth.middleware');

// Multer para aceptar CSVs
const storage = multer.diskStorage( opts: {
    destination: (req, file, cb) => cb(null, 'uploads/'),
    filename: (req, file, cb) => cb(null, Date.now() + path.extname(file.originalname))
});
const upload = multer( options: { storage });

router.get('/form_csv', ensureAuthenticated, checkRole( roles: [1]), importarController.vistaFormularioImportacion);

// AJAX: obtener plantillas del vendedor
router.get('/plantillas/:id_vendedor', ensureAuthenticated, checkRole( roles: [1]), importarController.obtenerPlantillasPorVendedor);

// POST: subir y procesar el archivo
router.post('/procesar_csv', ensureAuthenticated, checkRole( roles: [1]), upload.single( name: 'archivo_csv'), importarController.importarCSV);

module.exports = router;
```

```
const pool = require('../db');
const fs = require('fs');
const path = require('path');
const csv = require('csv-parser');

exports.vistaFormularioImportacion = async (req, res) => { ... };

exports.obtenerPlantillasPorVendedor = async (req, res) => { ... };

exports.importarCSV = async (req, res) => { ... };
```