Proyecto J&A Select

Ariana Rodriguez Morales

Luis Ignacio Salas Salazar

Álvaro Rivas Galagarza

UNIVERSIDAD FIDELITAS

Desarrollo de Aplicaciones Web y Patrones

San José, Costa Rica

Abstract – This project involved the development of a web-based e-commerce platform for the entrepreneurial initiative J&A Select, following a structured planning process that incorporated software development methodologies and user-centered design principles. The goal was to create a virtual store capable of offering a seamless shopping experience while addressing the specific needs of both customers and administrators of the system.

To achieve this, requirements were identified through the analysis of user stories, which defined the main functionalities such as product visualization, category filtering, shopping cart management, secure payments, order tracking, and customer support. Based on these requirements, the system architecture was designed using the MVC (Model-View-Controller) pattern, enabling a clear separation of responsibilities and ensuring scalability.

The implementation was carried out using Spring Boot as the main framework, Thymeleaf for dynamic views, and MySQL as the database engine, complemented with Spring Security to manage authentication and role-based access control. The design process also included the creation of wireframes and prototypes for each of the web sections (Home, Store, Contact, About Us, Policies), ensuring consistency and usability across the platform.

The result is the successful development of J&A Select, a secure, reliable, and user-friendly e-commerce website that allows customers to browse and purchase products while providing administrators with the necessary tools to manage the online store effectively.

I. Introducción

El presente trabajo consiste en el desarrollo de J&A Select, una aplicación web orientada al comercio electrónico que busca ofrecer a los usuarios una experiencia de compra moderna, intuitiva y segura. El proyecto surge como respuesta a la necesidad de los pequeños y medianos emprendimientos de contar con plataformas digitales que permitan exhibir y comercializar sus productos de manera eficiente, alcanzando así a un público más amplio en un entorno altamente competitivo.

J&A Select se plantea como una tienda virtual donde los clientes pueden navegar entre diferentes categorías de productos como: accesorios gaming, belleza y salud, artículos para el hogar y más, visualizar imágenes y descripciones detalladas, y realizar compras en línea con procesos simplificados y métodos de pago seguros. La propuesta incluye funcionalidades clave como la gestión de un carrito de compras, filtrado de productos por categoría, registro de usuarios, notificaciones de pedidos y un módulo de administración para mantener actualizada la oferta disponible.

La solución fue implementada bajo el modelo de arquitectura MVC (Model-View-Controller) utilizando Spring Boot como framework principal, Thymeleaf para la generación de vistas dinámicas, y MySQL como sistema de gestión de base de datos. Además, se integraron mecanismos de seguridad con Spring Security para garantizar la protección de datos y el control de acceso según roles.

De esta manera, J&A Select no solo busca cumplir con las historias de usuario planteadas, sino también convertirse en una herramienta escalable y confiable que aporte valor a los emprendedores y a los clientes finales, fortaleciendo la digitalización del comercio en el contexto actual.

II. Historias de Usuario

1. **Visualización de productos.**

   Como cliente interesado en productos únicos, quiero ver imágenes y descripción de los productos, para asegurarme de que cumple con mis expectativas antes de comprar.

2. **Filtrado por categoría.**

   Como usuario del sitio Web, quiero filtrar los productos por categorías como “Accesorios Gaming” o “Belleza y Salud”, para encontrar rápidamente lo que estoy buscando.

3. **Proceso de Compra sencillo.**

Como comprador, quiero poder agregar productos a un carrito y finalizar mi compra fácilmente, para tener una experiencia rápida y sin complicaciones.

4. **Métodos de pagos seguros.**

   Como cliente, quiero contar con varios métodos de pago seguro (SINPE Móvil), para sentirme confiado al realizar mis compras.

5. **Seguimientos de pedidos.**

   Como comprador frecuente, quiero recibir notificaciones a o tener un área donde pueda revisar el estado de mi pedido, para saber cuándo llegará mi producto.

6. **Atención al Cliente.**

   Como usuario con dudas, quiero acceder fácilmente a un chat o sección de contacto, para recibir soporte cuando lo necesite.

7. **Versión Móvil optimizada.**

   Como visitante desde mi celular, quiero que la página se vea bien y funcione correctamente en dispositivos móviles, para poder comprar desde cualquier lugar.

8. **Registro y beneficios**

   Como usuario recurrente, quiero poder registrarme y recibir promociones exclusivas, para obtener beneficios por ser cliente fiel.

9. **Añadir Productos**

   Como trabajador de J&A Select, quiero añadir nuevos productos al sistema para mantener actualizada la oferta en la tienda en línea

10. **Cambiar estado de disponibilidad del Producto.**

    Como administrador, quiero cambiar el estado de los productos (disponible, agotado, oculto) para que los clientes solo vean lo que pueden comprar

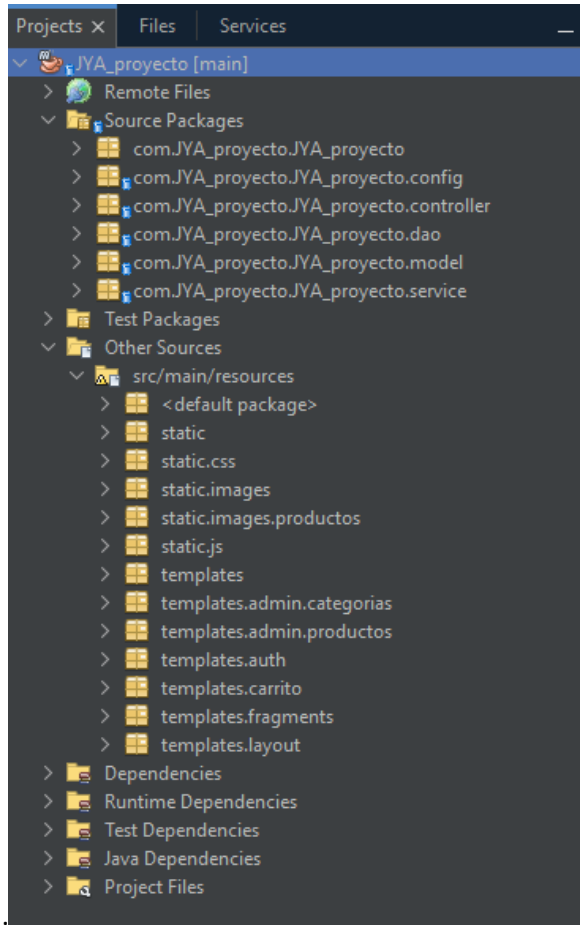
III. Desarrollo

El desarrollo sigue el patrón **MVC (Model–View–Controller)**.

- En la carpeta **src/main/java** (equivalente a *Source Packages* en NetBeans) se encuentra la lógica de negocio: clases, servicios e interfaces en Java que permiten la funcionalidad de la tienda.
- En **src/main/resources/templates** (equivalente a *Other Sources*) se ubican las vistas HTML de cada sección del sitio, construidas con Thymeleaf.
- En **src/main/resources/static** se almacenan los recursos estáticos: imágenes, hojas de estilo CSS, librerías JavaScript y fragmentos reutilizables (como *header* y *footer*), los cuales se integran en todas las páginas.

Para el diseño visual y la adaptabilidad en distintos dispositivos, el proyecto utiliza el framework Bootstrap, que facilita la creación de un sitio moderno, responsivo y consistente.

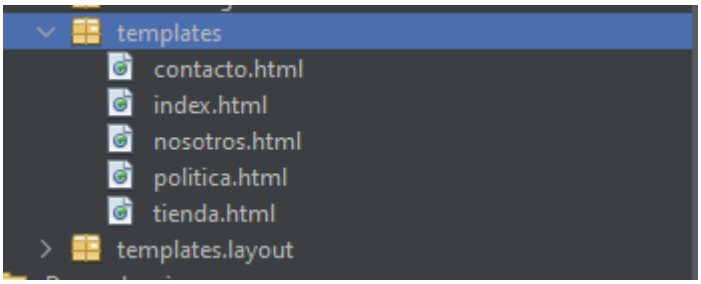
A continuación, se muestra una imagen de la estructura de carpetas dentro de Apache NetBeans.



Como parte del diseño de *J&A Select*, el sitio web se divide en cinco secciones principales:

1. Inicio.html
2. Tienda.html
3. Nosotros.html
4. Contacto.html
5. Política.html

Consulte la captura de pantalla a continuación para mayor referencia.



Cada una de las secciones del sitio web está basada en su respectivo diseño previamente elaborado con la herramienta Penpot, lo que garantiza consistencia visual y una estructura bien definida. Como se mencionó anteriormente, los componentes del header y el footer han sido codificados de forma independiente y se encuentran dentro de la carpeta **templates/layout**, con el objetivo de reutilizarlos en todas las páginas y mantener uniformidad en la navegación.

Asimismo, el proyecto incluye una vista de **Login** destinada al administrador, que permite iniciar sesión de manera segura para gestionar los productos de la tienda virtual (agregar, modificar y actualizar), reflejando los cambios en la **base de datos**. Esta funcionalidad está respaldada por la capa de backend en Java/Spring Boot, organizada en entidades, repositorios (DAO), servicios y controladores:

- **Entidades (model/domain):** Usuario, Rol, Producto, Categoria, Carrito/CarritoRegistro, MensajeContacto (mapeadas con JPA/Hibernate).
- **Repositorios (DAO):** UsuarioDao, ProductoDao, CategoriaDao, CarritoRegistroDao *(ej. consultas como findAllByUsuario y operaciones deleteRow)*, MensajeContactoDao.
- **Servicios (service):** UsuarioService, ProductoService, CategoriaService, CarritoService y el servicio de autenticación UsuarioDetailsServiceImpl *(implementa UserDetailsService para integrar credenciales y roles con Spring Security)*.
- **Controladores (controller):** AuthController *(gestiona /login y sesiones)*, ProductoController/TiendaController *(listado, detalle y CRUD de productos)*, CategoriaController, CarritoController, y ContactoController *(por ejemplo, manejo de formularios y vistas Thymeleaf)*.
- **Seguridad (config):** SecurityConfig *(configura rutas públicas/privadas, acceso por rol y encoder de contraseñas)*.

Con esta arquitectura, las vistas Thymeleaf en templates/ consumen los datos expuestos por los controladores; los controladores delegan la lógica a los servicios; y los servicios operan sobre los repositorios JPA para persistir los cambios en la base de datos. De esta forma, el administrador puede autenticarse, gestionar el catálogo y ver los cambios reflejados de inmediato en la tienda.

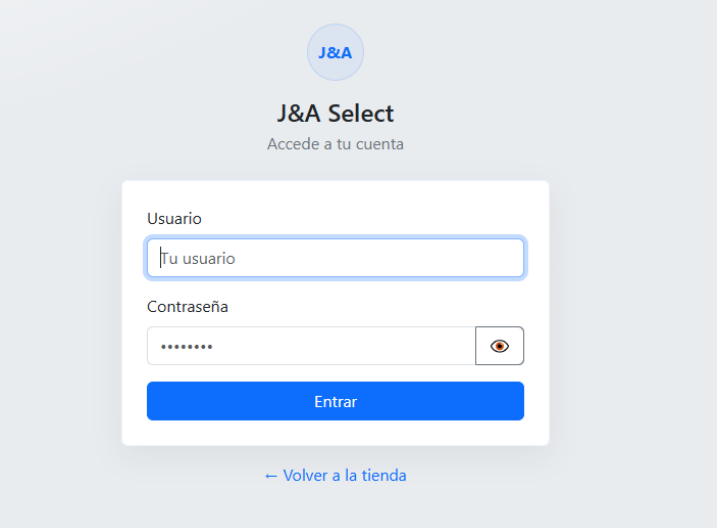


Imagen de la vista del login.html.

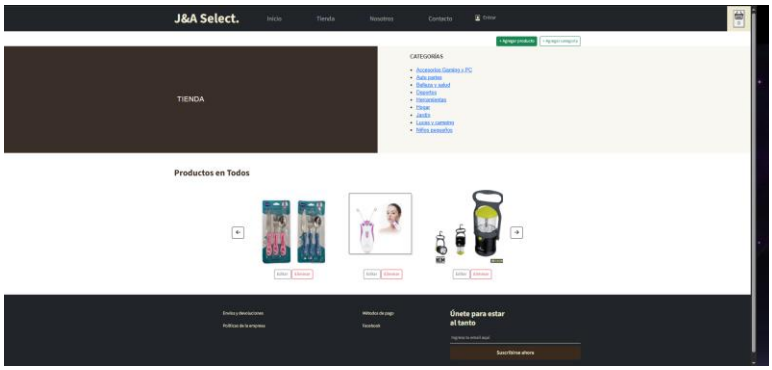
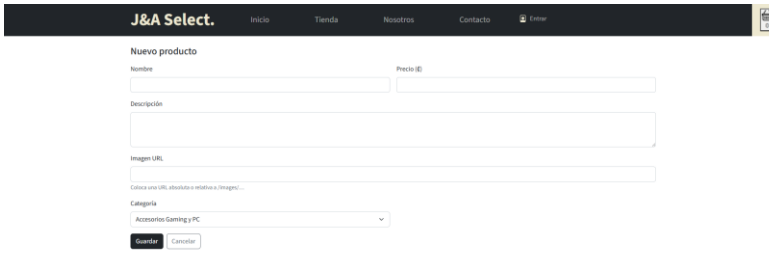


Imagen de la vista de tienda.html con los botones que gestionan el CRUD del Rol Admin.

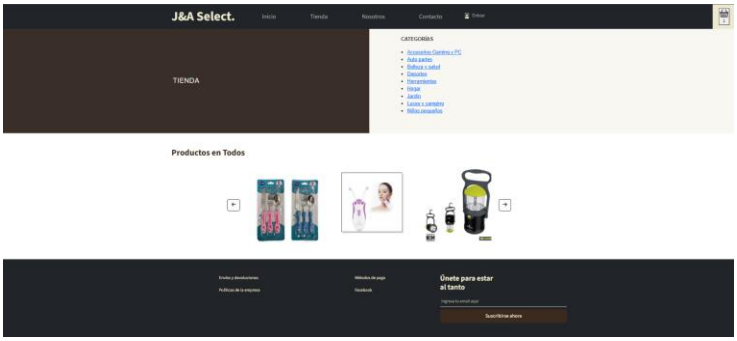


Vista del form.html para agregar nuevos productos como Admin.

Vista del form.html para agregar nuevas categorías como Admin.

Tienda.html

La sección tienda.html corresponde al núcleo de la aplicación, ya que en ella se muestran los productos disponibles dentro de la tienda virtual. Esta vista está construida con Thymeleaf y se encarga de renderizar dinámicamente la información proveniente de la base de datos, incluyendo el nombre, precio, descripción e imagen de cada producto. Para lograr esta integración, se utilizan distintas capas en Java/Spring Boot: la entidad Producto (que modela los datos del catálogo), el repositorio ProductoDao (que permite realizar consultas a la base de datos), el servicio ProductoService (que contiene la lógica para obtener, filtrar o manipular los productos) y el controlador TiendaController (que gestiona las peticiones HTTP y envía la lista de productos a la vista). De esta manera, cualquier usuario sea cliente o administrador puede visualizar el catálogo en tiempo real, garantizando que los cambios realizados en la base de datos se reflejen automáticamente en la interfaz de la tienda virtual.



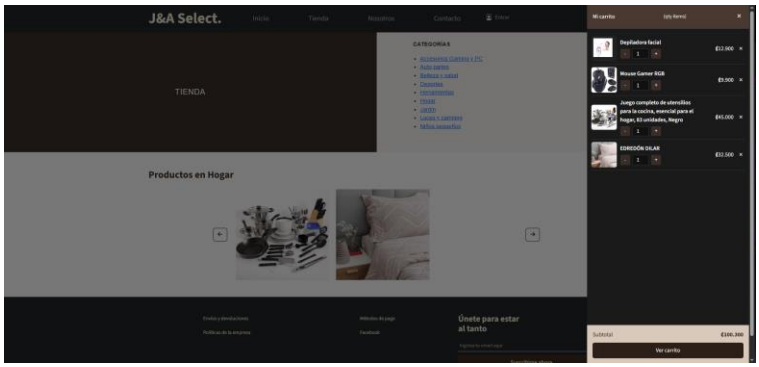
Vista de Tienda.html.

Carrito y Proceso de compra

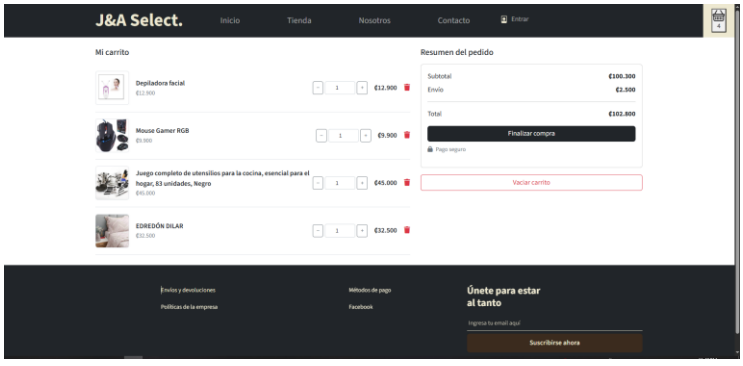
En tienda.html, al hacer clic sobre un producto se abre un modal de previsualización (Thymeleaf + JS) que muestra nombre, precio, descripción e imagen. Los datos del modal provienen de la entidad Producto, expuesta por el TiendaController (que delega en ProductoService y este a su vez en ProductoDao). El modal incluye un botón “Agregar al carrito” que dispara una petición (POST/Fetch) hacia el CarritoController, el cual utiliza CarritoService para crear/actualizar un CarritoRegistro asociado al usuario y persistirlo vía CarritoRegistroDao (por ejemplo, localizando la fila con findRow(uid, pid) o listando con findAllByUsuario(uid)). Tras agregar, se actualiza el navbar del carrito (contador y total) mediante un fragmento Thymeleaf que recalcula el desglose (subtotal por ítem y total), lógica que reside en CarritoService. El detalle completo del carrito se renderiza en la plantilla templates/carrito/pagina.html, donde se itera la colección de CarritoRegistro para mostrar cantidades, precios unitarios y totales parciales. Finalmente, cuando el usuario confirma la

acción de compra simulada (sin flujo de checkout), se muestra un popup de confirmación (JS) que informa “compra realizada” y opcionalmente limpia/actualiza la vista; esta confirmación se dispara desde el CarritoController tras completar la operación en CarritoService y persiste el estado en CarritoRegistroDao. Con este esquema, Producto, ProductoDao, ProductoService, TiendaController, CarritoRegistro, CarritoRegistroDao, CarritoService y CarritoController coordinan la previsualización, el agregado al carrito, la actualización del navbar y el desglose mostrado en **/carrito/pagina.html**, además del popup de confirmación de compra.

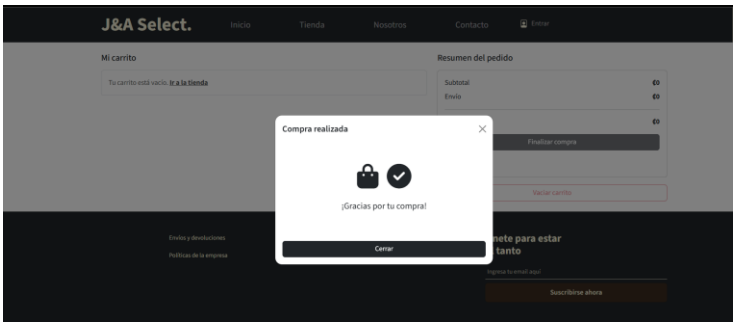
Vista del popup para previsualización de productos.



Vista del navbar del carrito.



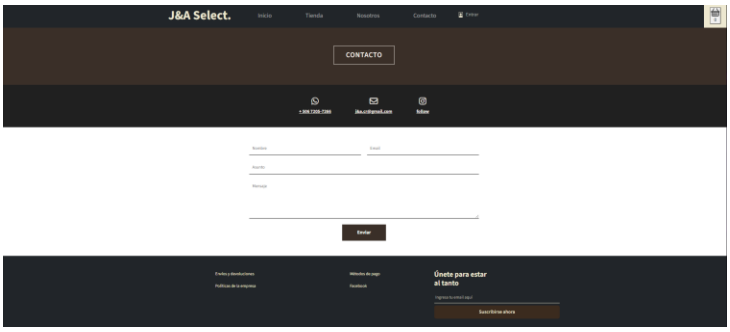
Vista final del carrito con resumen del pedido.



Vista de compra realizada satisfactoriamente.

Contacto.html

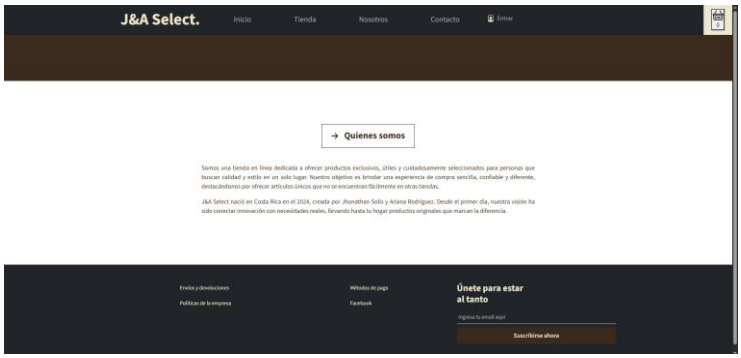
La sección contacto.html permite a cualquier usuario enviar un mensaje a través de un formulario con campos como nombre, correo, asunto y contenido. Al enviarlo, los datos son procesados por el ContactoController, que recibe la petición y la delega al MensajeContactoService. Este servicio crea una instancia de la entidad MensajeContacto, que modela el mensaje con sus atributos (id, nombre, email, asunto, contenido y fecha de envío), y la persiste en la base de datos utilizando el repositorio MensajeContactoDao. Gracias a este flujo, cada interacción desde contacto.html queda almacenada en la base de datos, asegurando que los administradores puedan consultar posteriormente los mensajes recibidos y mantener un historial de comunicación con los clientes.



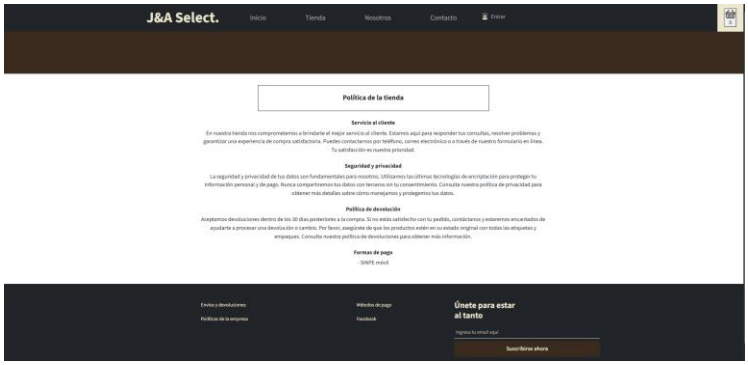
Vista de contacto.html

Nosotros.html & Politica.html

Las vistas nosotros.html y politica.html corresponden a secciones meramente informativas y estáticas, cuyo propósito es brindar contenido adicional al usuario acerca de la tienda y sus políticas. A diferencia de otras páginas del proyecto, estas no requieren interacción con la capa de Java ni con la base de datos, ya que no contienen lógica dinámica. No obstante, están construidas con HTML, CSS y el framework Bootstrap, lo que asegura una apariencia uniforme y responsiva en relación con el resto del sitio web. Además, comparten los mismos fragmentos reutilizables, como el header y el footer, lo que contribuye a mantener la coherencia visual y de navegación en toda la aplicación.



Vista de nosotros.html



Vista de políticas.html

Configuración y arranque

- **JyaProyectoApplication**: clase principal que inicia la aplicación Spring Boot.
- **SecurityConfig**: define reglas de seguridad, rutas públicas y privadas, e integra la autenticación de usuarios.
- **SeedUsersConfig**: inserta usuarios y roles iniciales en la base de datos para pruebas y configuración básica.

```java
public class SecurityConfig {

    private final UserDetailsService uds;

    public SecurityConfig(@Qualifier("userDetailsService") UserDetailsService uds) {
        this.uds = uds;
    }

    @Bean
    public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }

    @Bean
    public AuthenticationProvider authenticationProvider(PasswordEncoder enc) {
        DaoAuthenticationProvider p = new DaoAuthenticationProvider();
        p.setUserDetailsService(uds);
        p.setPasswordEncoder(enc);
        return p;
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http,
                                                   AuthenticationProvider ap) throws Exception {
        http
          .authenticationProvider(ap) // forzar provider
          .authorizeHttpRequests(auth -> auth
              .requestMatchers("/", "/index", "/tienda/**", "/contacto", "/nosotros", "/politica",
                               "/css/**", "/js/**", "/images/**", "/webjars/**", "/login", "/registro/**")
              .permitAll()
              .requestMatchers("/carrito/**", "/checkout/**", "/comprar/**").hasAnyRole("USER","ADMIN")
              .requestMatchers("/admin/**", "/productos/**").hasRole("ADMIN")
              .anyRequest().authenticated()
          )
          .formLogin(f -> f
              .loginPage("/login")
              .usernameParameter("username")
              .passwordParameter("password")
              .defaultSuccessUrl("/", true)
              .failureUrl("/login?error")
              .permitAll()
          )
          .logout(l -> l.logoutUrl("/logout").logoutSuccessUrl("/").permitAll());

        return http.build();
```

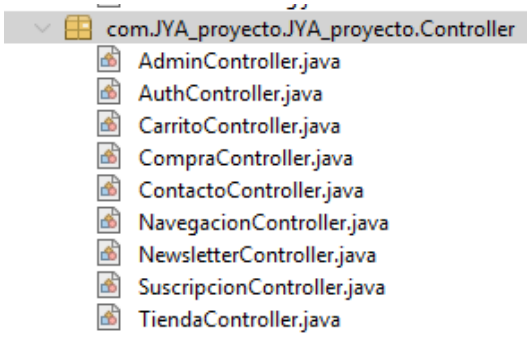
```java
@Configuration
public class SeedUsersConfig {
//ES UN SOLO ARCHIVO PARA PRUEBAS DE USUARIOS!!
    @Bean
    CommandLineRunner seedAndCheck(UsuarioDao dao, PasswordEncoder enc) {
        return args -> {
            // --- ADMIN ---
            Usuario admin = dao.findByUsername("admin");
            if (admin == null) {
                admin = new Usuario();
                admin.setUsername("admin");
                admin.setNombre("Admin");
                admin.setApellidos("Site");
                admin.setCorreo("admin@site.test");
                admin.setActivo(true);
                admin.setPassword(enc.encode("admin123"));
                Rol ra = new Rol();
                ra.setNombre("ROLE_ADMIN");
                ra.setUsuario(admin);
                admin.setRoles(new ArrayList<>(List.of(ra)));
                dao.save(admin);
                System.out.println(">> Creado admin/admin123");
            } else {
                // fuerza password y rol correctos
                admin.setPassword(enc.encode("admin123"));
                ensureRole(admin, "ROLE_ADMIN");
                dao.save(admin);
                System.out.println(">> Actualizado admin/admin123");
            }

            // --- USUARIO1 ---
            Usuario u1 = dao.findByUsername("usuario1");
            if (u1 == null) {
                u1 = new Usuario();
                u1.setUsername("usuario1");
                u1.setNombre("Usuario");
                u1.setApellidos("Normal");
```

Controladores (Controllers)

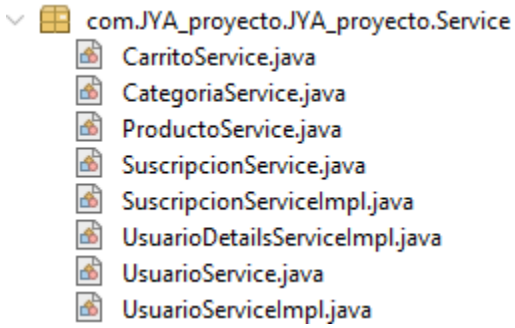
- **NavegacionController**: maneja la navegación general del sitio (index, contacto, nosotros, tienda, política).

- **CarritoController**: gestiona el carrito de compras; permite agregar, actualizar, eliminar productos y calcular totales.
- **CompraController**: controla el proceso de compra, mostrando productos, calculando impuestos y costos de envío.
- **TiendaController**: muestra los productos disponibles y permite aplicar filtros por categoría.
- **AuthController**: gestiona login, registro y autenticación de usuarios.
- **AdminController**: centraliza las operaciones de administración (gestión de productos, usuarios, roles).
- **NewsletterController**: permite suscribir usuarios al boletín informativo.
- **SuscripcionController**: maneja el flujo de planes de suscripción.
- **ContactoController**: procesa el formulario de contacto y almacena mensajes de usuarios.



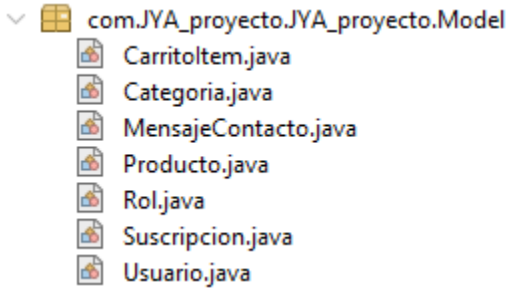
Servicios (Services)

- **CarritoService**: lógica del carrito: ver productos, actualizar cantidades, eliminar ítems, calcular subtotales y total.
- **ProductoService**: lógica de gestión de productos (consultas, alta, baja, actualización).
- **CategoriaService**: gestiona las categorías de productos.
- **UsuarioService / UsuarioServiceImpl**: lógica para registro, búsqueda y administración de usuarios.
- **SuscripcionService / SuscripcionServiceImpl**: maneja los planes de suscripción.
- **UsuarioDetailsServiceImpl**: conecta el sistema de usuarios con Spring Security para la autenticación.



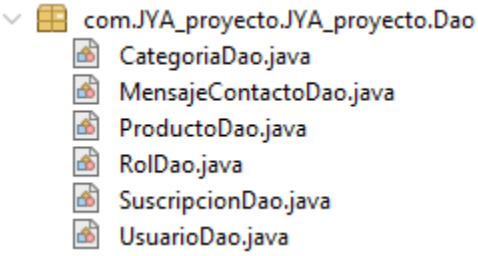
Modelos / Entidades (Model)

- **Producto**: representa un producto con nombre, descripción, precio e imagen.
- **CarritoItem**: objeto que representa un producto en el carrito con su cantidad.
- **Categoria**: categoría de producto (ej. belleza, accesorios, gaming).
- **Usuario**: datos de los usuarios registrados en la tienda.
- **Rol**: define los roles (ADMIN, CLIENTE).
- **Suscripcion**: representa planes de suscripción disponibles.
- **MensajeContacto**: entidad que guarda los mensajes enviados desde el formulario de contacto.



DAO (Data Access Objects)

- **ProductoDao, CategoriaDao, UsuarioDao, RolDao, SuscripcionDao, MensajeContactoDao**: son repositorios que conectan las entidades con la base de datos, permitiendo operaciones CRUD sin necesidad de SQL manual.



IV. Conclusión

En conclusión, el proyecto J&A Select representa la construcción de una tienda virtual desarrollada bajo el patrón MVC con Spring Boot, Thymeleaf y JPA/Hibernate, que integra de manera efectiva las capas de modelo, servicio y controlador para garantizar una correcta separación de responsabilidades. A lo largo de su implementación se estructuraron vistas dinámicas, como tienda.html y carrito/pagina.html, respaldadas por entidades, repositorios y servicios en Java que permiten la gestión del catálogo de productos, el carrito de compras y el envío de mensajes de contacto a la base de datos. Del mismo modo, se incorporaron vistas estáticas como nosotros.html y politica.html, las cuales, aunque no requieren lógica de backend, mantienen la coherencia visual gracias al uso de Bootstrap, HTML y CSS. Con esta arquitectura, el proyecto ofrece una base sólida y escalable para el comercio electrónico, asegurando tanto la funcionalidad técnica como una experiencia de usuario consistente y clara, además de sentar las bases para futuras mejoras, como la implementación de un flujo de checkout completo y un panel administrativo más robusto.