

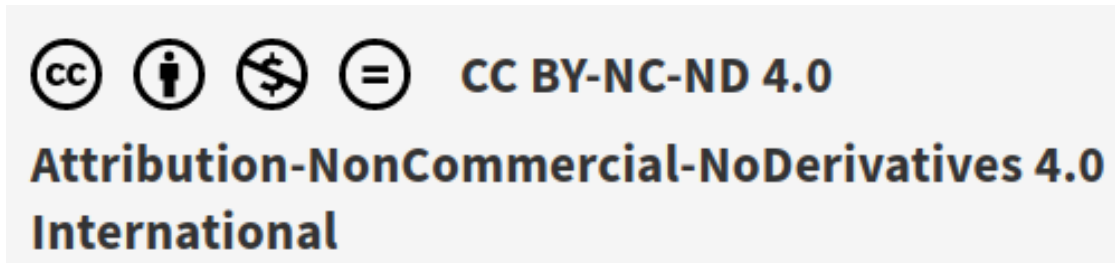
# UrbanVibe

Balbino Moyano López

Junio 2024



Este documento se encuentra bajo una licencia Creative Commons de Atribución-CompartirIgual (CC BY-SA).



Atri  
buci  
ón-  
Com  
parti  
rlgu

al (CC BY-SA)

Esto significa que puedes:

- Compartir: copiar y redistribuir el material en cualquier medio o formato.
- Adaptar: remezclar, transformar y construir sobre el material para cualquier propósito, incluso comercialmente.

Bajo las siguientes condiciones:

- Atribución: debes dar crédito de manera adecuada, proporcionar un enlace a la licencia e indicar si se han realizado cambios. Puedes hacerlo de cualquier manera razonable, pero no de una manera que sugiera que el licenciante te respalda a ti o al uso que hagas del trabajo.
- Compartir igual: si remezclas, transformas o creas a partir del material, debes distribuir tus contribuciones bajo la misma licencia que el original.

Para más detalles, consulta la [licencia completa](#).

# Índice

- Introducción
  - Datos del proyecto
  - Planificación
  - Ejecución del proyecto
  - Organización del proyecto
- Tecnologías
- Analisis de Diagramas
  - Diagrama clases
  - Diagrama de la base de datos
  - Diagrama de ayuda para la lógica
- Implementación
- Conclusiones
  - Experiencias personales y pequeñas explicaciones
- Bibliografía y Webgrafía

## Introducción

El proyecto consiste en una página de ropa urbana que es lo que más a día de hoy está de moda, donde hay ropa tanto de mujer como de hombre, y todo diseñado a mano desde 0 con varias ideas proyectadas en mi cabeza y sobre todo con muchas ganas y dedicación y horas de trabajo en una de las etapas más duras que he vivido, y estoy muy contento de haber conseguido lo que he conseguido con este proyecto a pesar de las adversidades.

Este proyecto surgió de una idea propia la cual a mi siempre me ha gustado que es la de crear yo mi propia tienda de ropa con los productos que yo mismo he diseñado y poder hacer una venta de lo mismos.

Otra de las principales ideas del proyecto sería una vez finalizado si es posible poder llegar a vender el software a alguna empresa que esté a punto de empezar en este mundo y necesite una tienda virtual, como digo es una idea ya luego al final hay que pulir muchas cosas y muchos detalles.

El logo de la tienda principalmente es el siguiente:



Si nos ponemos a analizar el logo podemos ver como la U hace referencia a una asa de una bolsa o esa es la intención que la U simule lo que es una asa de una bolsa y debajo podemos ver el nombre de la empresa junto a su eslogan, principalmente este sería el diseño que llevarían las bolsas.

## Datos del proyecto

Nombre	Apellidos	Título	Ciclo	Año	Centro educativo
Balbino	Moyano López	Desarrollo de Aplicaciones Multimedia	Superior	2024	IES Virgen del Carmen

## Planificación

Esta sería aproximadamente la planificación del proyecto:

- 1ª Semana
  - En la primera semana de trabajo he esquematizado todo el proyecto, he buscado otras ideas de proyecto, además de hacer un diagrama UML de lo que a mi me gustaría que fuera finalmente este proyecto.
  - He decidido hacer el proyecto en spring debido a que he notado un importante interés en esta tecnología por las empresas que actualmente buscan desarrolladores de este estilo.
  - Esta etapa fue indispensable para crear unas buenas bases para el proyecto (UrbanVibe), el haber hecho los diagramas y haberlo esquematizado todo me ha ayudado muchísimo en la elaboración del proyecto porque gracias a todos estos recursos he podido entender mejor como iban a interrelacionar las clases entre sí y lo que yo iba a necesitar para hacer que todo funcionara correctamente.
- 2ª Semana
  - He intentado profundizar en una planificación detallada en el proyecto, definiendo tareas y una serie de objetivos intentando que sean unos plazos realistas como sería por ejemplo objetivos semanales
  - También intenté hacerme un boceto de la estructura que iba a tener la base de datos para así tener una idea más clara de como se iba a almacenar y organizar la información de dicho proyecto.
  - He estado investigando páginas similares: Pull&Bear, Bershka, Zara, Mango...

Todo esta investigación me sirve a mi para tener varias ideas sobre como puedo hacer una interfaz vistosa, simple, atractiva y sobre todo intuitiva, en definitiva y para resumir, podemos decir que una lluvia de ideas para mi
- 3ª Semana
  - He estado investigando más estilos para mi futura página de ropa además de estar mirando la posibilidad de añadir un bot en python para simular una atención al cliente

- He estado estudiando diferentes maneras de contactarme con el usuario con el rol cliente como por ejemplo mensajería via gmail,etc...
  - Esta semana fue una de las semanas marcadas por una cantidad razonable de horas investigando sobre nuevas tecnologías y herramientas para enriquecer y hacer más fácil la experiencia al usuario con el rol Cliente que en este caso yo lo he llamado "Customer"
- 4º Semana
  - He decidio hacer el proyecto con Spring-MVC por que lo veo a pesar de haber estado debatiendo entre hacer Spring de backend y usar React en el frontend,se me sugirió hacerlo entero con Spring y en un futuro ya usaré React para el frontend.
  - Al no usar react uso mucho javascript para algunos estilos como por ejemplo el de la página principal del endpoint "/clothes" que voy pasando cada x tiempo una foto distinta
- 5º Semana
  - Estoy terminando el backend y realizando varios ajustes en cuanto al frontend para hacerlo más visual e intuitivo.
  - Todos estos ajustes los realizo porque me he tirado muchas pero muchísimas horas buscando en distintas páginas de ropa para ir copiando estilos de una y de otra porque pienso que es lo más estético y lo que más llama la atención incluso lo que más fácil puede llegar a ser para el usuario.
- 6º Semana
  - Primera versión demo semifuncional
  - Primera revisión de la documentación para ver que estén todos los puntos necesarios.
  - En la primera versión semifuncional,se puede hacer un pedido correctamente además de ver de forma detalla la ropa,controlo varias excepciones como por ejemplo sino se selecciona una talla que te ponga;"Por favor selecciona una talla"
  - Además también controlo varios errores de login y registro y de manera visual se lo enseño al cliente
- 7º Semana
  - Resultados finales (proyecto terminado: tutorial, aplicación...)
  - Segunda revisión del documento donde ya estén todos los apartados necesarios
  - Preparación de la presentación
  - Preparando javadoc y toda la documentación adicional del proyecto además de videos explicativos etc...
- 8º Semana

- Pulimos los posibles “bugs”
- Corrección de algunos códigos poco legibles también he intentado afinar un poco el código para que sea lo menos posible código denominado “spaguetti”
- Entrega del documento final
- 9ª Semana
  - Organización de la presentación
  - Entrega de la presentación para la exposición
- 10ª Semana
  - Presentación del proyecto

## Ejecución del proyecto

Para ejecutar el proyecto realizado en spring primeramente necesitaremos haber montado previamente docker y haberlo ejecutado, para ello deberíamos de irnos a la carpeta donde se aloja nuestro docker-compose.yml y deberemos de ejecutar el siguiente comando:

```
docker-compose -f nombre_del_archivo.yml up -d
```

Si ya tenemos el docker creado de antes nos aparecerá esto:

```
C:\Users\balta\Desktop\UrbanVibe\UrbanVibe\stack>docker-compose -f docker-compose.yml up -d
[+] Running 2/2
 - Container stack-db-1      Started           1.4s
 - Container stack-adminer-1 Started           1.4s
```

En cambio si es la primera vez que lo hacemos tendremos esto:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.4412]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\balta\Desktop\UrbanVibe\UrbanVibe\stack>docker-compose -f docker-compose.yml up -d
[+] Running 3/3
 - Network stack_skynet      Created           0.2s
 - Container stack-adminer-1 Started           6.5s
 - Container stack-db-1      Started           6.6s

C:\Users\balta\Desktop\UrbanVibe\UrbanVibe\stack>
```

Con este comando se nos ejecutará en los puertos que nosotros tengamos predefinidos en el archivo.yml el adminer, es decir, nuestra gestor de la base datos, para ver nuestra base de datos en ejecución entonces necesitaremos ejecutar spring que para ello podemos hacerlo de varias maneras, o desde visual studio con la extensión de spring que hay un botón de ejecutar o directamente con el comando:

```
mvn spring-boot:run
```

Una vez ejecutemos el comando esto:

```
flb2oxrb4.argfile' 'com.shop.iesvdc.shop.ShopApplication'

  ____  _
 / ___|| | | |
| |___| |_| |
 \___|_||_|_|_|

:: Spring Boot ::      (v3.2.3)

2024-06-11T21:20:42.933+02:00 INFO 12172 --- [shop] [ restartedMain] com.shop.iesvdc.shop.ShopApplication : Starting ShopApplication using Java
17.0.9 with PID 12172 (C:\Users\balta\Desktop\UrbanVibe\UrbanVibe\target\classes started by balta in C:\Users\balta\Desktop\UrbanVibe\UrbanVibe)
2024-06-11T21:20:42.941+02:00 INFO 12172 --- [shop] [ restartedMain] com.shop.iesvdc.shop.ShopApplication : No active profile set, falling back
to 1 default profile: "default"
```

Ejecutaremos el proyecto en el puerto por defecto que usa spring el 8080, si quisieramos cambiar el puerto por defecto de spring, lo deberemos de hacer en el archivo application.properties y poner algo talque así:

```
server.port=7070
```

He puesto el puerto 7070, pero se podría poner otro sin problemas siempre y cuando no esté ocupado

## Organización del proyecto

La organización del proyecto está compuesta por las siguientes carpetas:

- docs: aquí se alojan todas las fotos o documentos de interés que necesitemos.
- src:
  - main:
    - conf: aquí se alojan todo tipo de configuraciones relacionadas con la seguridad, login, register....
    - controller: aquí se alojan todo tipo de controladores con el que manejamos su crud completo de nuestras clases modelo
    - model: nuestras clases entidad y las que le dan sentido al proyecto
    - repo: se alojan todas las interfaces necesarias
    - service: aquí pondremos si fuera necesario los servicios de nuestras clases modelo
    - ShopApplication.java: Es el archivo principal de nuestro proyecto, sin él, no ejecutaría
  - resources: se encuentra el código de básicamente todo el frontend que uso:
    - static: aquí se encuentra todas las imágenes, etc.



- application.properties: se encuentran las contraseñas necesarias para que se conecte con la base de datos, no se debería de subir a git por motivos obvios.
- import.sql: se encuentra todos los insert si queremos añadir algun dato en alguna tabla directamente en la base de datos.
- templates: aquí se encuentran todas las plantillas.
  - users: en el encontramos los archivos:
    - users.html: muestra todos los usuarios, si tienes los requisitos suficientes para verlos(Admin), además de poder borrar y editar los usuarios.
    - add.html: muestra el formulario para añadir un usuario, si tienes los requisitos suficientes para verlos(Admin).
    - edit.html: muestra el formulario para editar un usuario, si tienes los requisitos suficientes para verlos(Admin).
  - help.html: al igual que acerca, es totalmente estético.
  - denegado.html: es un archivo el cual indica a un usuario sino tiene permisos pues lo redirigimos a la siguiente página.
  - menu.html: es el principal endpoint al cuál se le redirige a un usuario que tiene el rol Customer en el cual el usuario puede elegir que tipo de ropa quiere ver, si la de hombre, mujer o toda la ropa.
  - error.html: en caso de que exista algún tipo de error en nuestra página lo redirigimos a está que resulta más cómodo para el cliente ver que hay un error y que tiene que contactar con soporte.
  - index.html: es la página a la que redirigimos por defecto una vez hecho el login.
  - login.html: es la página por defecto de nuestro login.
  - signup.html: es la página para hacer un registro.
- stack: aquí se encuentran los archivos:
  - docker-compose.yml: este archivo es nuestro contenedor y dentro montamos las imagenes que necesitamos y el motor de base de datos que usamos

- .env: aquí encontramos las contraseñas de nuestro docker-compose
- setup.sql: es un archivo que sirve para inicializar la base de datos con el nombre que nosotros queramos ponerle a nuestra base de datos.
- pom.xml: es nuestro archivo de dependencias, sin él, el proyecto sería inservible

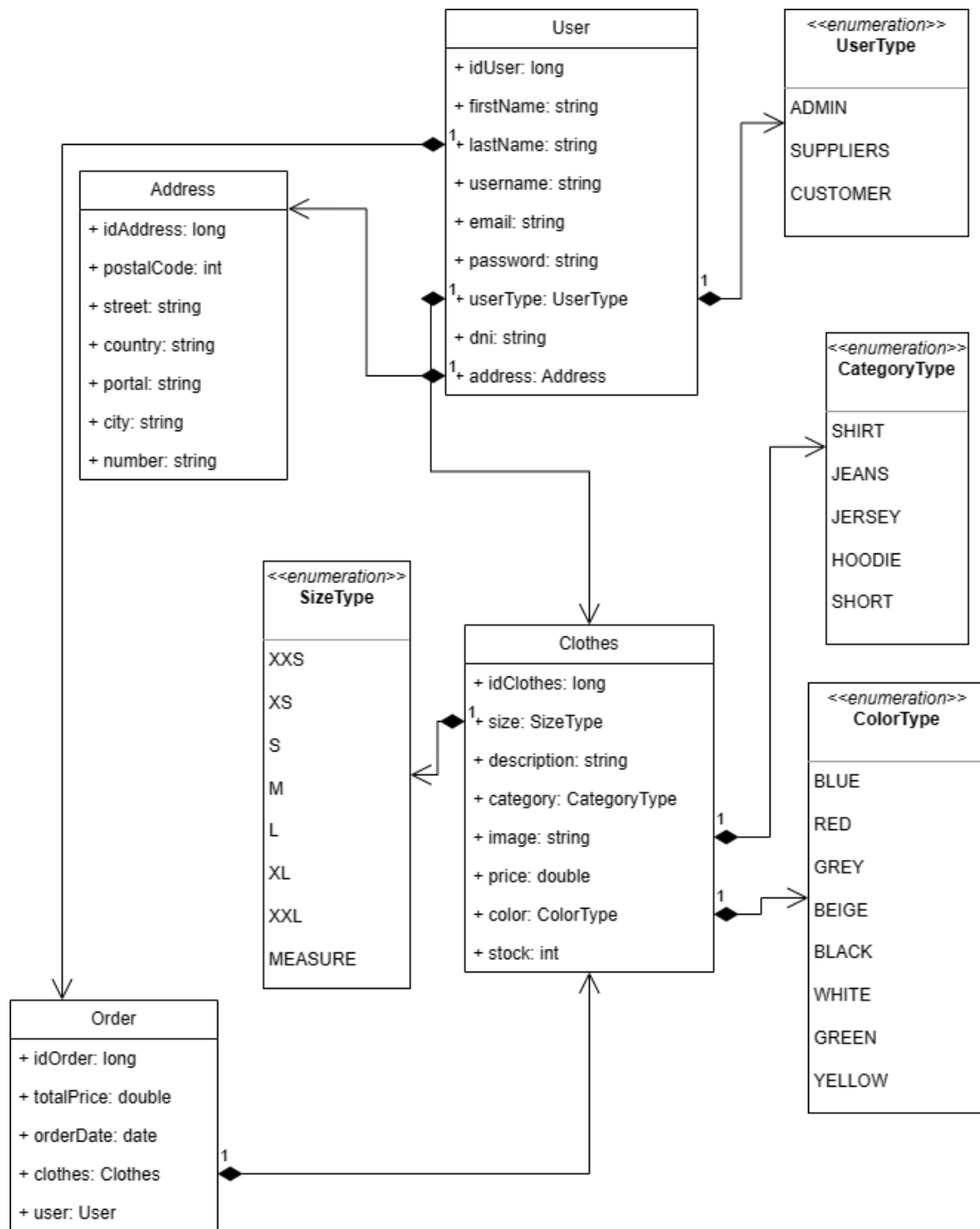
## Tecnologías

Las tecnologías que uso son las siguientes:

- 1º Spring MVC: he usado el modelo-vista-controlador, por que veía muy interesante lo que podía llegar a lograr.
- 2º Webjars: lo he usado principalmente para algunos iconos que tiene mi aplicación.
- 3º Docker: lo he usado principalmente para ver como se vería mi proyecto en diferentes ordenadores para intentar hacer el proyecto lo más responsive posible y sobre todo para luego poder publicarlo en un servidor.
- 4º Lombok: principalmente lo he usado para las anotaciones @Data, @Entity, @NoArgsConstructor y ahorrarme getters, setter, constructores y hacer mi código mucho más limpio y para hacer la base de datos automáticamente una vez yo lance el proyecto en spring
- 5º Servidor: finalmente no se si haré un deploy en alguna página pero la idea es que si hago un deploy o sea en la página de **IONOS, GitHub Page...** o algunas páginas de las que he mencionado anteriormente

## Análisis

**Diagrama de mi aplicación para la tienda UrbanVibe.**



### Diagrama

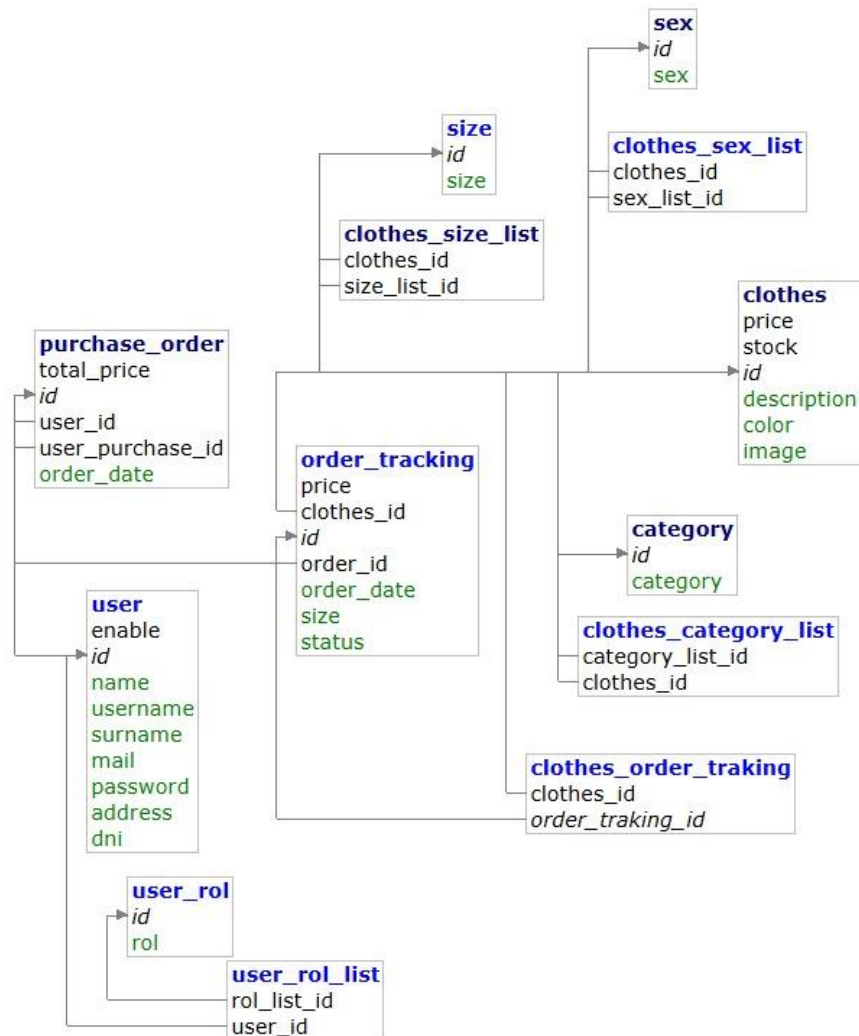
En este diagrama podemos ver las relaciones que he usado, aunque finalmente he cambiado varias cosas por ejemplo el color no es un enum sino que es un String y por ejemplo también uso una entidad adicional la cual se llama OrderTracking que tiene de relación: \*

PurchaseOrder: 1 pedido tiene varias líneas de pedido la cual contiene mucha ropa. \* Clothes: 1 prenda de ropa puede estar en muchos pedidos.

Gracias a esta relación puedo ver la ropa que tiene cada pedido en el backend tal y como enseño en los videos.

---

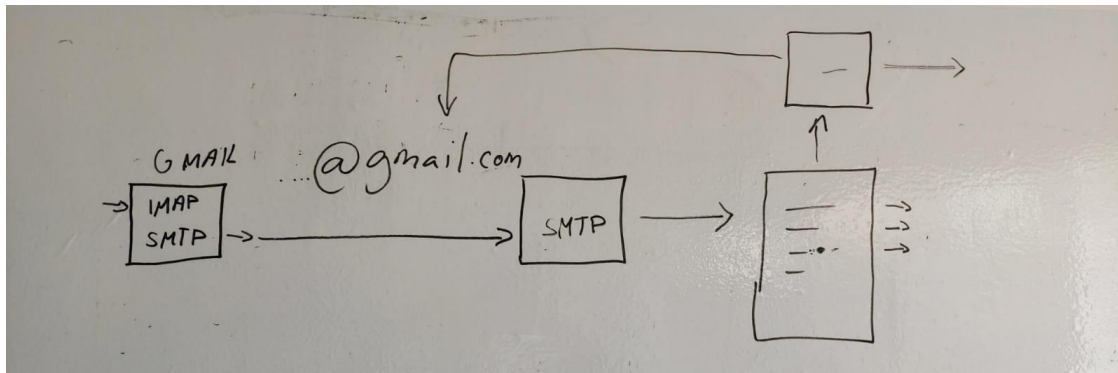
### Esquema de la base de datos.



Aquí muestro mi esquema de la base de datos donde podemos ver todos los id y relaciones de cada una de las tablas que conforman la tienda de ropa de UrbanVibe

---

Esquema que me ha ayudado a realizar la lógica de gmail.



Este es el principal esquema que me ha ayudado a realizar la lógica de mandar un email, que he explicado más detalladamente en el siguiente apartado de implementación

## Implementación

Explicación de como he hecho la lógica para mandar un gmail:

### MailConfig.java

Necesitamos un config para indicar a spring la configuración que vamos a usar, en este caso lo hacemos con la anotación, @Bean indicándole que puede encontrar el valor de estas variables en nuestro archivo properties con el nombre correspondiente, además usamos los métodos de la clase javaMail que previamente están importados para poder usarlos, donde indicamos puerto, correo, protocolo(smtp), credencial del correo que vamos a usar.

@Configuration

```
public class MailConfig {
```

```
    @Value("${spring.mail.host}")
    private String host;
```

```
    @Value("${spring.mail.username}")
    private String mail;
```

```
    @Value("${spring.mail.port}")
    private int port;
```

```
    @Value("${spring.mail.password}")
    private String password;
```

@Bean

```
public JavaMailSender mailSender() {
    JavaMailSenderImpl mailSender = new JavaMailSenderImpl();
    mailSender.setHost(host);
    mailSender.setPort(port);
    mailSender.setUsername(mail);
```

```

        mailSender.setPassword(password);

        Properties props = mailSender.getJavaMailProperties();
        props.put("mail.transport.protocol", "smtp");
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.debug", "true");

        return mailSender;
    }
}

```

## MailController.java

Necesitamos un controlador para indicar el endpoint al que vamos a dirigir al usuario para así poder hacer la petición al servidor de gmail y este nos de el okey para poder mandar el correo a la persona correspondiente

```

@RestController
@RequestMapping("/help")
public class MailController {

    @Autowired
    private MailService emailService;

    @PostMapping("/send")
    public ResponseEntity<?> receiveRequestEmail(@RequestBody MailDTO
mailDTO){

        System.out.println("Mensaje recibido: "+ mailDTO);

        emailService.sendMail(mailDTO.getTo(), mailDTO.getSubject(),
mailDTO.getMessage());
        Map<String,String> response = new HashMap<>();
        response.put("estado","Enviado");

        return (ResponseEntity<?>) ResponseEntity.ok(response);
    }

}

```

## MailDTO.java

Al nosotros mandar el correo, lo mandamos en formato json y necesitamos de alguna manera pasar nuestros datos del formulario a json, para ello necesitamos esta clase con la que pasaremos nuestros datos a json

```

@Getter
@NoArgsConstructor
@AllArgsConstructor
public class MailDTO {
    private String[] to;
    private String subject;
    private String message;
}

```

## MailRepo.java

Interfaz que controla el envío de correo

```

public interface MailRepo {
    void sendMail(String[] to,String subject,String message);
}

```

## MailService.java

Clase que implementa la interfaz de MailRepo y que es la que se encarga de enviar el correo

```

@Service
public class MailService implements MailRepo{

    @Value("${spring.mail.username}")
    private String from;

    @Autowired
    private JavaMailSender mailSender;

    @Override
    public void sendMail(String[] to, String subject, String message) {
        SimpleMailMessage mailMessage = new SimpleMailMessage();

        mailMessage.setFrom(from);
        mailMessage.setTo("Correo que recibe el mensaje");
        mailMessage.setSubject(subject);
        mailMessage.setText(message);
        mailSender.send(mailMessage);
    }
}

```

## Mandar email desde el html

Necesitamos hacer un formulario y desde el script recoger los datos con el dom y una vez recogidos hacer una petición para poder mandar el correo

```

<script src="https://smtpjs.com/v3/smtp.js"></script>

<script>
  const enviarCorreo = (event) => {
    event.preventDefault(); // Evita que se envíe el formulario
    directamente

    const name = document.getElementById('name').value;
    const phone = document.getElementById('phone').value;
    const asunto = document.getElementById('subject').value;
    const mail = document.getElementById('mail').value;
    const message = document.getElementById('message').value;

    const mailTo = {
      to: ["Correo que recibe el mensaje"],
      subject: asunto,
      message: `Nombre: ${name}\nTeléfono: ${phone}\nEmail:
${mail}\nMensaje: ${message}`
    };

    fetch('/help/send', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(mailTo),
    })
    .then(response => {
      if (response.ok) {
        alert('Correo enviado correctamente');
      } else {
        alert('Error al enviar el correo');
      }
    })
    .catch(error => {
      console.error('Error al enviar la solicitud:', error);
      alert('Error al enviar el correo');
    });
  };
</script>

```

## application.properties

```

spring.mail.host=dominio
spring.mail.port=puerto_que_usa_gmail
spring.mail.username=persona_enviar_correo
spring.mail.password=password
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true

```



---

Para hacer la lógica de mandar el correo una vez el pedido ha terminado, uso el siguiente método:

```
public void sendMailToLoggedInUser(String subject, String message) {
    Authentication authentication =
    SecurityContextHolder.getContext().getAuthentication();

    if (authentication != null && authentication.isAuthenticated()) {
        /*
         * Es un método que tiene la clase Authentication de donde saco
         el Objeto el cuál está actualmente Logueado
        */
        Object principal = authentication.getPrincipal();
        String email = null;

        if (principal instanceof UserDetails) {
            email = ((UserDetails) principal).getUsername();
        } else if (principal instanceof String) {
            email = principal.toString();
        }

        if (email != null) {
            sendMail(new String[]{email}, subject, message);
        }
    }
}
```

Con este código lo que obtengo es que el usuario el cuál está haciendo el pedido, es decir está logueado, recojo su email, compruebo si es nulo o no y con la ayuda de un instanceof de UserDetails, puedo obtener el nombre del usuario y con ello puedo obtener el email del usuario y paso el email a string y lo mando todo lo importante del pedido.

## Conclusiones

La conclusiones que he sacado con este proyecto son:

- 1º : A pesar del poco tiempo que he tenido lo he podido sacar adelante y he ido sacando todos los problemas que me han ido surgiendo y eso hace que esté realmente orgulloso de lo que he conseguido, aprendido y lo que he disfrutado haciendo el proyecto y poco a poco viendo avances del mismo
- 2º : Con tiempo dedicación y sobre todo sabiendo buscar y el dónde se pueden sacar muchas cosas que yo pensaba que no iba a poder llegar a sacar pero al fin y al cabo todo son horas y dedicación, todo este proyecto tiene mas de 50h ya que no me han pesado

hecharle las horas porque me gusta programar,es algo que me tranquiliza(depends del momento) y que a veces cuando salen las cosas suele ser muy gratificante la verdad.

## Experiencias personales y pequeñas explicaciones

En cuanto a experiencias personales,aquí voy a explicar como he desarrollado en su mayoría cada una de las interfaces; \* menu.html: principalmente esta interfaz me gustó mucho de una de las interfaces que tiene el berhska en la cual se componen de un div con 3 imagenes muy representativas de lo que te puedes llegar a encontrar cuando tu pinchas en ellas,la principal diferencia por ejemplo entre berhska y urbanvibe sería que los textos que ellos tienen como “Hombre”, “Mujer”,etc...

Tal que así:



*Bershka*

Y UrbanVibe tiene los textos de esta otra manera:

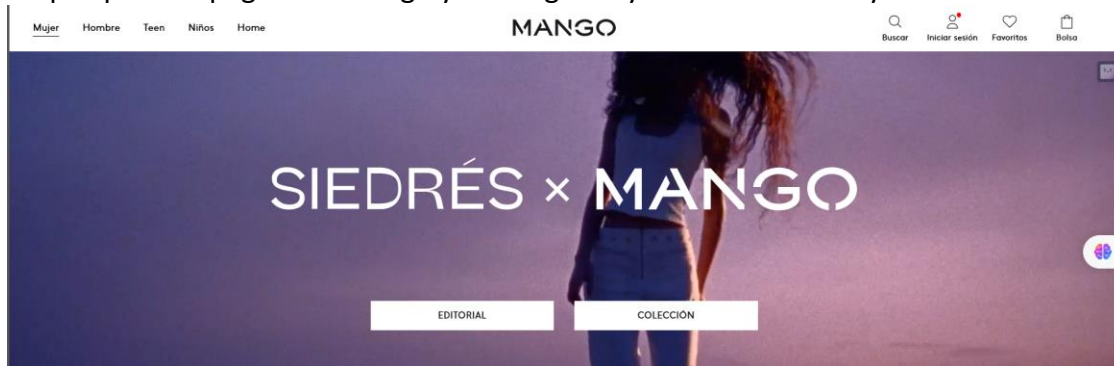
### URBANVIBE



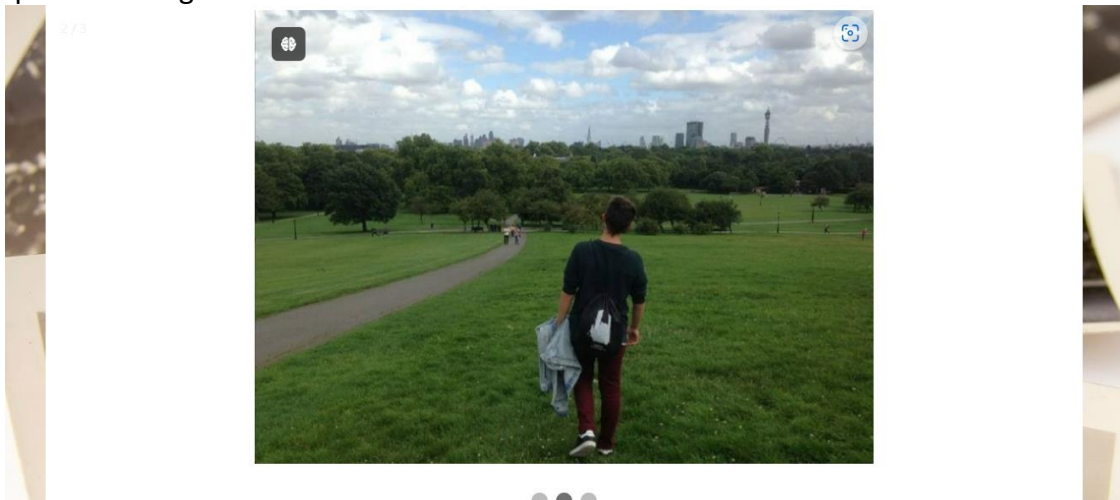
*UrbanVibe*

Además de implementarle un efecto que cada vez que está el ratón por encima de ampliación de la imagen.

- clothes.html: en este archivo la principal idea era hacer un menu lateral a la izuierda con un buscado a la derecha y un banner en el top de la página,que finalmente no ha sido así porque vi la página de mango y lo vi algo muy sencillo intuitivo y sobre todo estético;



Como podemos ver el menu consta de la ropa de hombre,mujer además de niño y de Home,muy parecida también a la que tiene Zara,pero yo en este caso solo tengo ropa de hombre y de mujer así ue he preferido usar hombre y mujer y otra para nuestra nueva colección donde aparece toda la ropa,además de yo meterle un toque personal fijandome en una de las páginas que hice hace muchos años donde tenía div completo que cada x segundo se cambiaba la foto:



Y el resultado obtenido de mezclar estas ideas es algo talque así:



Por ejemplo para hacer que cada x segundo se vaya cambiando la foto he usado este código:

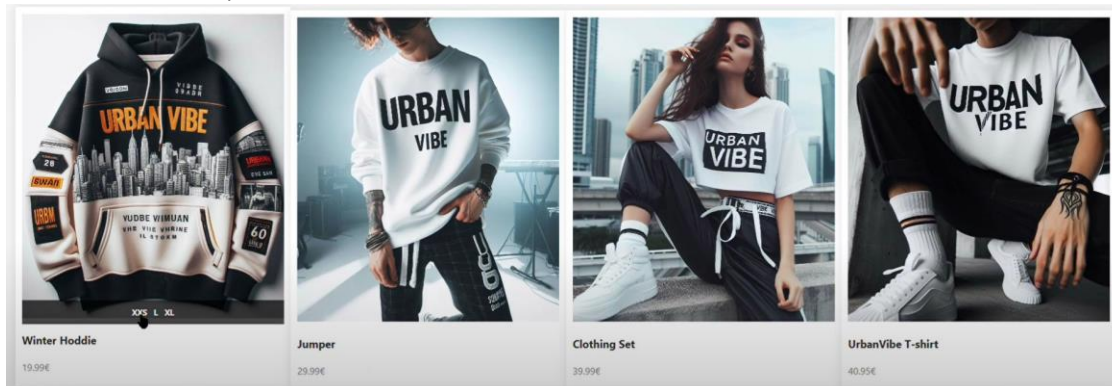
```
const heroBanner = document.getElementById('hero-banner');
const images = [
  '/img/mainimagen.jpeg',
  '/img/mainimagen2.jpeg',
  '/img/mainimagen3.jpeg',
  '/img/streetphoto.jpg',
  '/img/streetphoto2.jpg'
];
let currentIndex = 0;

function changeBackgroundImage() {
  currentIndex = (currentIndex + 1) % images.length;
  heroBanner.style.backgroundImage =
    `url(${images[currentIndex]})`;
}

setInterval(changeBackgroundImage, 5000);
```

Primeramente añado todas las imágenes que yo deseo que estén en una lista con respectiva ruta, indico que foto va a ser la primera y con el método `changeBackgroundImage()` y después de haber recogido el div con el DOM que se llama `hero-banner`, recorro la lista de uno en uno y le meto un intervalo de 5 segundos.

La forma de mostrar la ropa también tiene su aquel aunque al final no deja de ser un div de `width 100%` y que cada div que muestra una prenda ocupe un 25% de ese `width` y ya luego la altura la deseada, en UrbanVibe se muestran así:



Si nos fijamos en el video que tengo, podemos ver como tiene un `hover` que en cuanto tu pones el ratón encima del div, aparece las tallas que tiene disponibles, las cuales si tu pulsas en una de las tallas que aparece, automáticamente se añade al carrito en cambio si tu haces click en cualquier otro lugar que no sea donde se encuentran las tallas, se redirige a la siguiente página

También en este endpoint podemos ver 2 ventanas modales; una de ellas se despliega cuando hacemos click en el símbolo del carrito, mientras otra de ella se muestra cuando hacemos click en el lápiz, esto indica que vamos a



cambiar la talla de la ropa;

Modal de carrito:

```
<div class="cart-sidebar" id="cart-sidebar">
  <div class="cart-header">
    <span class="cart-title">Cesta</span>
    <i class="fas fa-times close-cart"></i>
  </div>
  <ul class="cart-items"></ul>
  <button class="checkout-btn" onclick="checkCart()">Purchase
Order</button>
</div>
```

Y con este script abro el carrito manualmente:

```
cartIcon.addEventListener('click', () => {
  cartSidebar.classList.add('open');
  /** Renderizar el carrito cuando se abre */
  renderCartItems();
});
```

Además de abrirlo, si tiene artículos el carrito se renderiza las prendas que existan.

Modal de Edición:

```
<div class="modal" id="edit-modal">
  <div class="modal-content">
    <div class="cart-header">
      <span class="cart-title">Editar Artículo</span>
      <i class="fas fa-times close-cart" id="close-edit-modal"></i>
    </div>

    <div class="cart-items edit-item-details">
      <img id="edit-item-image" src="" alt="" class="edit-item-image">
      <div class="show-clothes">
        <form id="editClothesForm">
          <div class="form-group">
            <strong>
              <p id="description" class="form-control-
description"></p>
            </strong>
          </div>
          <div class="form-group">
            <p id="price" class="form-control-price"></p>
          </div>
          <div class="form-group">
            <label for="sizes">Sizes</label>
            <p id="sizes" class="form-control"></p>
            <!-- Mostrar todas las tallas aquí -->
          </div>
        </form>
      </div>
    </div>
  </div>
</div>
```

```

        </form>

        <div class="edit-buttons">
            <button id="save-changes-btn" class="buttons-edit-cancel">Guardar cambios</button>
            <button class="close-modal buttons-edit-cancel"
id="close-modal-exit">Cancelar</button>
        </div>
    </div>
</div>
</div>
</div>
</div>

```

El script que abre el modal de edición es el siguiente:

```

function openEditModal(id) {
    console.log("Opening modal with ID:", id);
    fetch(`/clothes/edit/${id}`)
        .then(response => {
            if (!response.ok) {
                throw new Error("Network response was not ok");
            }
            return response.json();
        })
        .then(data => {
            console.log("Opening modal with data:", data);

            const imageElement = document.getElementById('edit-item-
image');
            const descriptionElement =
document.getElementById('description');
            const sizeElement = document.getElementById('size');
            const priceElement = document.getElementById('price');
            const sizesParagraph = document.getElementById('sizes');

            if (imageElement) {
                imageElement.src = data.image;
            } else {
                console.error("Image element not found");
            }

            if (descriptionElement) {
                descriptionElement.innerText = data.description;
            } else {
                console.error("Description element not found");
            }

            if (sizeElement) {
                sizeElement.innerText = data.size;
            } else {

```

```

        console.error("Size element not found");
    }

    if (priceElement) {
        priceElement.innerText = `${data.price}€`;
    } else {
        console.error("Price element not found");
    }

    if (sizesParagraph) {
        sizesParagraph.innerHTML = '';

        const availableSizes = data.availableSizes || [];
        availableSizes.forEach(size => {
            const sizeSpan = document.createElement('span');
            sizeSpan.innerText = size;
            sizeSpan.classList.add('size-option');
            if (data.size && size === data.size) {
                sizeSpan.classList.add('selected-size');
            }

            sizeSpan.addEventListener('click', () => {
                document.querySelectorAll('.size-
option').forEach(span => {
                    span.classList.remove('selected-size');
                });
                sizeSpan.classList.add('selected-size');
                localStorage.setItem('selectedSize', size);
                /** Actualiza la talla */
                data.size = size;
            });

            sizesParagraph.appendChild(sizeSpan);
        });
    } else {
        console.error("Sizes paragraph element not found");
    }

    /** Muestra el modal de edición */
    document.getElementById('edit-modal').style.display =
'flex';

    /** Maneja el evento de guardar los cambios dentro del
modal de edición */
    saveChangesBtn.onclick = () => saveChanges(data.id,
data.size);

    })
    .catch(error => {

```

```

        console.error("Error fetching data:", error);
    });
}

```

Con este código lo que hago es ver a que prenda ha hecho click, comprobar si existen cada uno de sus atributos y una vez comprobados abrimos el modal y ejercemos la lógica de una edición de talla.

Y los estilos que he usado en esta página son los siguientes:

```

@import
url('https://fonts.googleapis.com/css2?family=Montserrat:wght@700&display=swa
p');
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f5f5f5;
    display: flex;
    flex-direction: column;
    min-height: 100vh;
    overflow-x: hidden;
}
.header {
    background-color: #000;
    color: #fff;
    padding: 20px 0;
    width: 100%;
    position: fixed;
    top: 0;
    left: 0;
    z-index: 1000;
    height: 5.5%;
}
.navbar {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 0 20px;
}
.nav-list {
    cursor: pointer;
    list-style: none;
    padding: 0;
    display: flex;
    align-items: center;
}
.nav-list li {
    margin-right: 20px;
}

```



```

}
.nav-list li a {
    text-decoration: none;
    color: #fff;
    font-weight: bold;
}
.company-name {
    font-family: 'Montserrat', sans-serif;
    font-size: 20px;
    text-align: center;
    margin-right: 10px;
    letter-spacing: 1px;
    color: #fff;
}
.navbar-icons {
    display: flex;
    align-items: center;
    position: relative;
}
.navbar-icons i {
    font-size: 20px;
    color: #fff;
    margin-right: 10px;
    cursor: pointer;
}
/**
.search-input {
    position: absolute;
    right: 0;
    top: 50%;
    transform: translateY(-50%);
    height: 35px;
    padding: 5px;
    border: none;
    border-radius: 5px;
    width: 0;
    opacity: 0;
    transition: width 0.4s ease, opacity 0.4s ease;
}
.search-input.active {
    width: 200px;
    opacity: 1;
}
*/
.main-content {
    flex: 1;
    padding: 20px;
    margin-top: 80px;
}
.hero-banner {

```

```
    background: #000;
    text-align: center;
    color: #fff;
    padding: 100px 0;
    background-size: cover;
    background-position: center;
    position: relative;
    overflow: hidden;
}
.hero-title {
    font-size: 48px;
    margin-bottom: 20px;
    position: relative;
    z-index: 2;
}
.btn-primary {
    background-color: #333;
    padding: 15px 30px;
    border-radius: 5px;
    text-decoration: none;
    color: #fff;
    font-weight: bold;
    transition: background-color 0.3s ease;
    position: relative;
    z-index: 2;
}
.btn-primary:hover {
    background-color: #555;
}
.featured-products {
    display: flex;
    flex-wrap: wrap;
    width: 100%;
    padding: 20px 0;
    margin: 0;
}
.product-card {
    background-color: #fff;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.1);
    text-align: left;
    transition: transform 0.3s;
    padding: 10px;
    width: 25%;
    box-sizing: border-box;
    position: relative;
    display: flex;
    flex-direction: column;
    justify-content: space-between;
}
.product-card:hover {
```

```

        transform: translateY(-5px);
    }
    .product-card:hover .size-container {
        opacity: 1;
    }
    .product-card img {
        width: 100%;
        height: 450px;
        object-fit: cover;
        margin-bottom: 5px;
    }
    .image-container {
        position: relative;
    }
    .size-container {
        display: flex;
        justify-content: center;
        gap: 10px;
        position: absolute;
        bottom: 0;
        left: 0;
        right: 0;
        background-color: rgba(0, 0, 0, 0.7);
        padding: 10px;
        opacity: 0;
        transition: opacity 0.3s ease;
    }
    .sizes {
        color: #fff;
        font-size: 12px;
        font-weight: bold;
        cursor: pointer;
    }
    .product-title {
        font-size: 16px;
        margin-bottom: 5px;
        color: #333;
    }
    .product-price {
        font-size: 14px;
        color: #888;
        margin-bottom: 10px;
    }
    .footer {
        background-color: #000;
        color: #fff;
        text-align: center;
        padding: 20px 0;
        margin-top: auto;
    }
}

```

```

.footer p {
    margin: 0;
}
@media (max-width: 768px) {
    .navbar {
        flex-direction: column;
        align-items: center;
    }
    .company-name {
        margin-bottom: 10px;
    }
    .navbar-icons {
        margin-top: 20px;
    }
    .navbar-icons i {
        margin-right: 8px;
    }
    .product-card {
        width: 100%;
    }
}
@media (max-width: 480px) {
    .hero-title {
        font-size: 36px;
    }
    .btn-primary {
        padding: 10px 20px;
    }
}

.checkout-btn {
    display: block;
    width: 80%;
    margin: 0 auto;
    padding: 15px;
    background: #333;
    color: #fff;
    text-align: center;
    border: none;
    border-radius: 5px;
    font-size: 18px;
    font-weight: bold;
    cursor: pointer;
    transition: background 0.3s ease, transform 0.3s ease;
}
.checkout-btn:hover {
    background: #555;
    transform: scale(1.05);
}
.marquee-container {

```

```

    position: sticky;
    bottom: 0;
    width: 100%;
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 5px 0;
    z-index: 1000;
    overflow: hidden;
}
.marquee {
    display: inline-block;
    white-space: nowrap;
    animation: marquee 10s linear infinite;
}
@keyframes marquee {
    0% { transform: translateX(100%); }
    100% { transform: translateX(-100%); }
}
.modal-overlay {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.7);
    display: flex;
    justify-content: center;
    align-items: center;
    z-index: 2000;
}
.modal {
    display: none;
    position: fixed;
    top: 0;
    right: 0;
    width: 300px;
    height: 100%;
    background: #fff;
    box-shadow: -2px 0 5px rgba(0, 0, 0, 0.2);
    z-index: 1001;
    transition: right 0.3s ease;
}

.modal.open {
    display: block;
    opacity: 1;
    transform: translate(-50%, -50%) scale(1);
}

```

```
.modal-content {
  display: flex;
  flex-direction: column;
  height: 100%;
  width: 100%;
  overflow-x: hidden;
}

.modal-header,
.modal-footer {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 15px;
  border-bottom: 1px solid #ddd;
}

.modal-header {
  border-bottom: 1px solid #ddd;
}

.modal-footer {
  border-top: 1px solid #ddd;
}

.modal-title {
  font-size: 24px;
  font-weight: bold;
  color: #333;
}

.close-modal {
  font-size: 24px;
  cursor: pointer;
  color: #888;
}

.modal-body {
  flex: 1;
  overflow-y: auto;
}

.modal-form-group {
  margin-bottom: 20px;
}

.modal-form-group label {
  font-size: 16px;
  font-weight: bold;
}
```

```

        margin-bottom: 5px;
        display: block;
        color: #333;
    }

    .modal-form-group input,
    .modal-form-group select {
        width: 100%;
        padding: 10px;
        border: 1px solid #ddd;
        border-radius: 5px;
        font-size: 16px;
    }

    .modal-form-group select {
        -webkit-appearance: none;
        -moz-appearance: none;
        appearance: none;
    }

    #edit-item-image {
        width: 45%;
        height: auto;
        max-height: 200px;
        object-fit: cover;
        border-radius: 10px;
        margin-right: 10px;
    }

    #edit-item-title {
        font-size: 20px;
        font-weight: bold;
        color: #333;
        margin-bottom: 5px;
    }

    #edit-item-price {
        font-size: 18px;
        color: #888;
        margin-bottom: 10px;
    }

    #save-changes-btn,
    #close-modal-exit {
        background-color: #333;
        color: #fff;
        margin-right: 10px;
        padding: 10px 20px;
        border: none;
    }

```

```
        border-radius: 5px;
        font-size: 16px;
        cursor: pointer;
        transition: background-color 0.3s ease;
    }

    #save-changes-btn:hover,
    #close-modal-exit:hover {
        background-color: #555;
    }

    .cart-sidebar {
        position: fixed;
        top: 0;
        right: -100%;
        width: 300px;
        height: 100%;
        background: #fff;
        box-shadow: -2px 0 5px rgba(0, 0, 0, 0.2);
        transition: right 0.3s ease;
        z-index: 1001;
        display: flex;
        flex-direction: column;
        justify-content: space-between;
    }

    .cart-sidebar.open {
        right: 0;
    }

    .cart-header {
        display: flex;
        justify-content: space-between;
        align-items: center;
        padding: 15px;
        border-bottom: 1px solid #ddd;
    }

    .cart-title {
        font-size: 24px;
        font-weight: bold;
        color: #333;
    }

    .close-cart {
        font-size: 24px;
        cursor: pointer;
    }
```



```
.cart-items {  
  list-style: none;  
  padding: 0;  
  margin: 0;  
  flex: 1;  
  overflow-y: auto;  
}  
  
.cart-item {  
  display: flex;  
  justify-content: space-between;  
  margin-bottom: 15px;  
  padding: 10px;  
  border-bottom: 1px solid #ddd;  
}  
  
.cart-item img {  
  width: 80px;  
  height: 80px;  
  object-fit: cover;  
  margin-right: 15px;  
}  
  
.cart-items.edit-item-details{  
  overflow-x: hidden;  
}  
  
.cart-item-title {  
  font-size: 18px;  
  margin: 0 0 5px 0;  
  color: #333;  
}  
  
.cart-item-price {  
  font-size: 16px;  
  color: #888;  
}  
  
.checkout-section {  
  padding: 15px;  
  border-top: 1px solid #ddd;  
}  
  
.checkout-button {  
  display: block;  
  width: 70%;  
  padding: 10px;  
  background-color: #333;
```

```

        color: #fff;
        border: none;
        cursor: pointer;
        font-size: 16px;
        transition: background-color 0.3s ease;
        text-align: center;
        text-decoration: none;
    }

    .checkout-button:hover {
        background-color: #555;
    }

    .edit-item-image {
        width: 50% !important; /** Ajusta el ancho de La imagen según sea
necesario */
        height: auto !important; /** Esto mantendrá La proporción de La
imagen */
        max-height: 200px !important; /** Establece una altura máxima
para La imagen */
        display: block !important; /** Asegura que La imagen se muestre
como un bloque */
        margin: 0 auto !important; /** Esto centra La imagen
horizontalmente */
    }

    .edit-item-image img {
        width: 50% !important; /** Ajusta el ancho de La imagen según sea
necesario */
        height: auto !important; /** Esto mantendrá La proporción de La
imagen */
        max-height: 200px !important; /** Establece una altura máxima
para La imagen */
        display: block !important; /** Asegura que La imagen se muestre
como un bloque */
        margin: 0 auto !important; /** Esto centra La imagen
horizontalmente */
    }

    .edit-item-details{
        display: block !important;
        justify-content: start;
        align-items: center;
        margin-top: 20px;/** Esto centra La imagen horizontalmente */

    }

    #edit-item-image{
        width: 45%;
        height: auto;
    }

```

```

        max-height: 200px;
        object-fit: cover;
        border-radius: 10px;
        margin-right: 10px; /** Esto centra la imagen horizontalmente */
    }

    #edit-item-title {
        font-size: 20px;
        font-weight: bold;
        color: #333;
        margin-bottom: 5px;
    }

    #edit-item-price {
        font-size: 18px;
        color: #888;
        margin-bottom: 10px;
    }

    #save-changes-btn,
    #close-modal-exit {
        background-color: #333;
        color: #fff;
        padding: 10px 20px; /** Asegura el mismo tamaño y relleno */
        border: none;
        border-radius: 5px;
        font-size: 16px;
        cursor: pointer;
        transition: background-color 0.3s ease;
    }

    #save-changes-btn:~hover,
    #close-modal-exit:~hover {
        background-color: #555;
    }

    .edith2 {
        font-size: 24px;
        font-weight: bold;
        color: #333;
    }

    .show-clothes{
        display: block;
        margin: 10px;
        margin-left: 15%;
    }

    .edit-buttons{
        display: flex;

```

```

        margin: 0 -2rem;
        align-items:center;

    }

    .buttons-edit-cancel{
        margin: 0.8rem 0.2rem 0.2rem !important;

    }

    .cart-counter {
        display: none; /** Ocultar el contador por defecto */
        font-family: 'Dancing Script', cursive;
        position: absolute;
        top: -10px;
        right: -10px;
        background-color: rgb(245, 245, 220); /** Color de fondo rojo */
        color: black;
        border-radius: 50%;
        padding: 5px 8px; /** Aumentar el relleno para que sea más
visible */
        font-size: 14px; /** Aumentar el tamaño de la fuente */
        font-weight: bold; /** Hacer el texto en negrita */
        box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2); /** Agregar sombra */
    }

    .cart-icon .cart-counter {
        display: inline; /** Mostrar el contador cuando se pasa el ratón
sobre el icono */
    }

    .form-control-description{
        font-size: 18px;
        color: #333;
    }

    .form-control-price{
        color:#888;
    }

    .form-control span {
        font-size: 1rem;
        padding: 15px 20px;
        border: 1px solid #ccc;
        border-radius: 100%;
        cursor: pointer;
        transition: all 0.3s;
    }

    .form-control .selected-size {

```

```

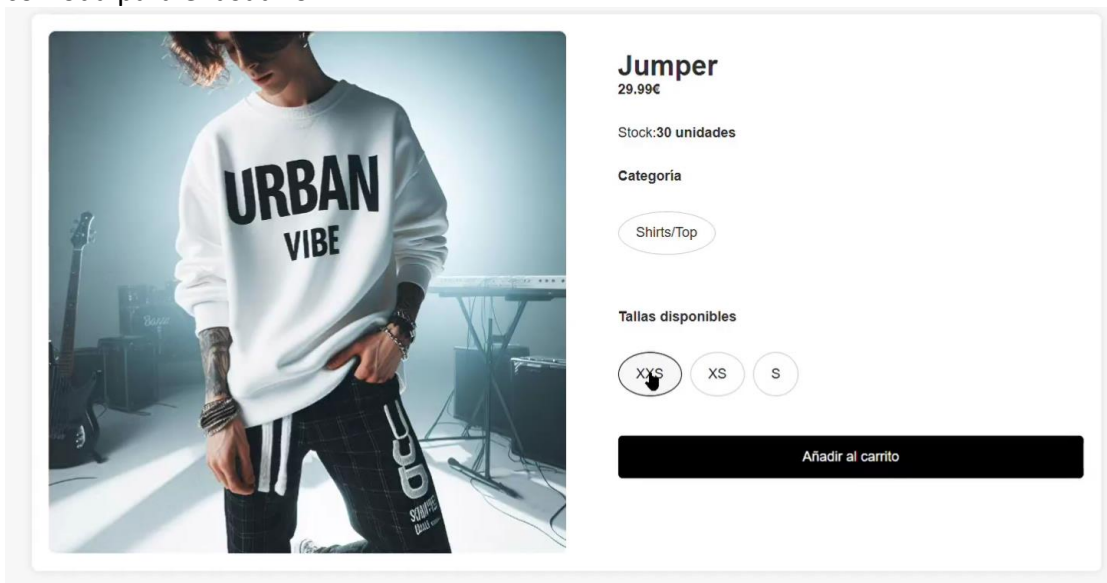
        background-color: black;
        color: white;
        flex-wrap: wrap;
    }

    .urban-contact{
        text-decoration:none;
        color:white;
        cursor: pointer;
    }

```

Estos estilos sirven para darle toda la estética a la página como por ejemplo los modales situados a la derecha de la página con z-index y muy importante el position:fixed para fijar la posición. Además de darle estilos a mis cardview y mi carrito.

- details.html: esta es una de las páginas encargadas de hacer más visual eh intuitivo los atributos de la ropa,es decir,se muestran de una manera mejor planteada y mas visible y cómoda para el usuario:



Y como podemos ver muestro el stock,el precio,la imagen,las tallas disponibles y categoría a la que pertenece,si le das a añadir al carrito sin antes haber seleccionado una talla,le aparecerá “Por favor seleccione una talla”, de color rojo para que el usuario cliente sepa en todo momento que es lo que está pasando,luego una vez seleccione la talla se añadirá al carrito con la talla que el usuario ha elegido.

Para manejar si el usuario hace click en la talla o en otro lugar he usado el siguiente código:

```

document.addEventListener('DOMContentLoaded', function() {
    document.addEventListener('click', function(event) {
        const target = event.target;
        const isSize = target.classList.contains('sizes');
    });
});

```

```

        if (isSize) {
            const clotheElement = target.closest('.product-card');
            const clotheId = clotheElement.dataset.id;
            const size = target.dataset.size;
            addToCart(clotheId, size);
        } else {
            const clotheElement = target.closest('.product-card');
            if (clotheElement) {
                const clotheId = clotheElement.dataset.id;
                window.location.href = `clothes/details/${clotheId}`;
            }
        }
    });
});

```

Con el DOM recogemos el valor que buscamos, en este caso, la “cardview” en la cual, si donde hace click el cliente es donde se encuentran las tallas entonces llamo a la función que me añade la ropa al carrito, y si no hace click en este lugar, entonces cojo el id de la ropa en donde haya hecho click y le redirecciono a su respectivo endpoint “*clothes/details/{clothes.id}*”

Y estos son sus respectivos estilos:

```

body {
    font-family: 'Arial', sans-serif;
    background-color: #f7f7f7;
    margin: 0;
    padding: 0;
    color: #333;
}

.details-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    padding: 20px;
}

.details-content {
    display: flex;
    justify-content: space-between;
    width: 100%;
    max-width: 1200px;
    background-color: #fff;
    padding: 20px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    border-radius: 8px;
    margin-top: 5rem;
}

```

```
.details-left {
  width: 50%;
  display: flex;
  justify-content: center;
  align-items: center;
}

.details-image {
  max-width: 100%;
  max-height: 600px;
  object-fit: cover;
  border-radius: 8px;
}

.details-right {
  width: 45%;
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.details-right h1 {
  font-size: 2rem;
  margin-bottom: 5px;
}

.details-right p {
  font-size: 1rem;
  margin-bottom: 5px;
}

.details-right ul {
  list-style-type: none;
  padding: 0;
  display: flex;
  gap: 10px;
  flex-wrap: wrap;
  margin-bottom: 5px;
}

.details-right ul li {
  font-size: 1rem;
  padding: 15px 20px;
  border: 1px solid #ccc;
  border-radius: 100%;
  cursor: pointer;
  transition: all 0.3s ease;
}
```

```

.details-right ul li:~hover {
    border-color: #000;
    background-color: #f7f7f7;
}

.details-right ul li.active {
    border-color: #000;
    background-color: #000;
    color: #fff;
}

.details-right button {
    padding: 15px;
    background-color: #000;
    color: #fff;
    border: none;
    cursor: pointer;
    font-size: 1rem;
    border-radius: 5px;
    transition: background-color 0.3s;
}

.details-right button:~hover {
    background-color: #333;
}

.added {
    background-color: green;
    cursor: default;
}

.subtitle {
    margin-top: -1rem;
}

.error-message {
    color: red;
    margin-bottom: 10px;
    display: none;
}

```


Estos estilos son muy importantes tanto por el error que le indico al usuario de un color llamativo para que sepa que tiene que añadir la talla como para que la talla se vea atractiva y la ropa que el usuario tiene intención de comprar se vea de una forma clara.

- orders.html: Para hacer esta ventana, creo que ha sido la menos costosa porque siempre he tenido en mente como quería que fuese ya que obviamente el cliente tiene que saber que tiene que hacer en todo momento y sobre todo ver la información de su



pedido, además obviamente como al registrarte, no metes todos los datos necesarios pues obviamente necesito un formulario que recoja todo lo posible en cuanto al usuario en cuestión, todo esto lo he pensado así para darl dinamismo a la aplicación y que no se hiciera pesada:


URBANVIBE



Winter Hoddie

Talla: XL


19.99€



Iphone XR Phonecase

Talla: TALLA UNICA


10€



Jumper

Talla: XXS

29.99€



Jumper

Talla: M

39.95€

Order resume

Name:

Surname:

DNI:

Address:

Total: 99.93 €

Cancel

Order

Recojo la ropa de la siguiente manera:

```

cartData.forEach((item) => {
    if (!item.id || isNaN(item.id)) {
        console.error("Invalid item in cart:", item);
        return;
    }

    const itemElement = document.createElement("div");
    itemElement.classList.add("order-item");

    itemElement.innerHTML = `
        
        <div class="details-clothes">
            <b><p>${item.description}</p></b>
            <p>Talla: ${item.size}</p>
        </div>
        <div class="details-price">
            <p>${item.price}€</p>
        </div>
    `;

    orderItemsContainer.appendChild(itemElement);
    totalPrice += parseFloat(item.price);
});

```

Aquí recojo los datos ue tengo almacenados del carrito del usuario y los muestro en pantalla de la misma manera y con los mismos estilos con la que muestro la ropa cuando se encuentra en el

carrito. Y ya una vez recogida la ropa con el formulario que tengo **ACTUALIZO** al usuario con ese mismo id y creo **UNA LINEA PEDIDO**, esto es muy importante porque no hay que confundir, necesitamos la línea pedido para ver que pedidos hay y dentro de estos pedidos que ropa se han llevado por esta razón es tan importante que creamos una línea pedido y no un pedido.

Y una vez se ha creado y hemos terminado la compra se nos redirigirá al siguiente endpoint `"/thnku"`

Los estilos que usa esta página son los siguientes:

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-color: #f4f4f4;
}

header nav {
  background-color: #333;
  color: white;
  padding: 1rem;
  text-align: center;
}

.checkout-container {
  max-width: 70%;
  margin: 2.5% auto;
  padding: 1rem;
  background-color: white;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1 {
  text-align: center;
}

.order-summary {
  margin-top: 2rem;
  display: flex;
  justify-content: space-between;
  height: 50%;
}

#order-items {
  min-width: 22rem;
}

.order-items {
```

```
    margin-bottom: 1rem;
    flex: 1;
}

.order-item {
    display: flex;
    justify-content: space-between;
    padding: 1rem 0;
    border-bottom: 1px solid #ddd;
    width: auto !important;
}

.order-item img {
    max-width: 100px;
    margin-right: 1rem;
}

.order-total {
    text-align: right;
    font-size: 1.2rem;
}

.form-container {
    flex: 1;
    padding: 1rem;
    background-color: #f9f9f9;
    border: 1px solid #ddd;
    margin-left: 0.2rem;
    max-width: 60%;
}

#checkout-button {
    display: block;
    width: 50%;
    padding: 1rem;
    background-color: #333;
    color: white;
    border: none;
    cursor: pointer;
    font-size: 1.2rem;
    margin: 1rem;
}

#checkout-button:hover {
    background-color: #555;
}

.cancel-btn{
```

```

    display: block;
    width: 50%;
    padding: 1rem;
    background-color: #333;
    color: white;
    border: none;
    cursor: pointer;
    font-size: 1.2rem;
    margin: 1rem;
  }
  .cancel-btn:hover{
    background-color: #555;
  }
}

.details-clothes {
  display: grid;
  align-items: left;
  text-align: left;
  width: 75%;
}
.details-price {
  width: 30%;
  margin-right: 2rem;
  margin: 0;
}

.details-price p{
  margin-right: 0.5rem;
}

.order-box {
  max-width: auto;
  height: 35rem;
  overflow-y: auto;
  min-width: 40%;
}

.form-container {
  flex: 1;
  padding: 1rem;
  background-color: #f9f9f9;
  border: 1px solid #ddd;
  margin-left: 0.2rem;
  max-width: 60%;
}

```

```

.form-group {
  margin-bottom: 1rem;
}

.form-group label {
  display: block;
  margin-bottom: 0.5rem;
  font-weight: bold;
}

.form-group input {
  width: calc(100% - 1rem);
  padding: 0.5rem;
  border: 1px solid #ddd;
  border-radius: 5px;
  font-size: 1rem;
}

.urban-title{
  font-size:30px;
  font-weight: bold;
  cursor: pointer;
}

.btns-action{
  display:flex;
  width:100%;
}

```

Gracias a estos estilos, le doy formato a mi formulario, a la forma de ver la ropa que se recoge del carrito del usuario.

- thnku.html: En este endpoint únicamente me sirve para dar las gracias al usuario por su compra y a mí me hace saber que si ha llegado a este endpoint también le ha llegado su respectivo correo indicando que la compra fue un éxito:

## URBANVIBE

### ¡Gracias por tu compra!

En UrbanVibe valoramos tu confianza. Esperamos que disfrutes de tu ropa, diseñada con pasión y estilo.

No dudes en ponerte en contacto con nosotros si necesitas ayuda o tienes alguna pregunta.

[Volver a la tienda](#)

### UrbanVibe

Y aquí está el correo verificando la compra:



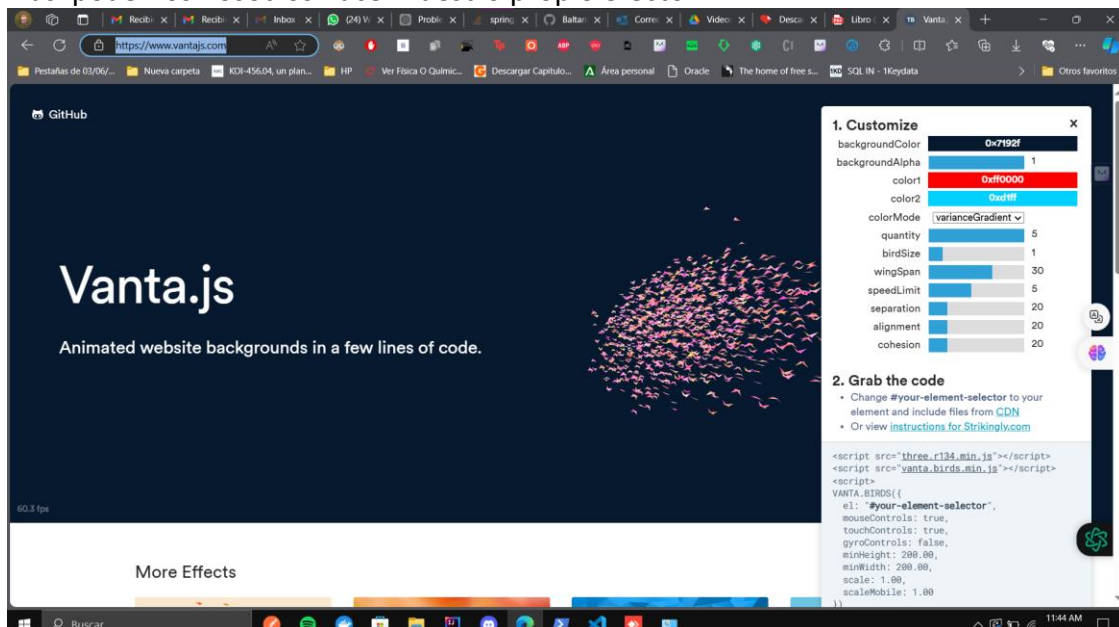
Otro efecto chulo que tengo al iniciar sesión con rol de Admin o al cerrar sesión sería el siguiente:



El cuál

está hecho en su mayoría con la documentación que tiene vanta en su propia web, ya yo únicamente le he dado los estilos que a mi me gustan y más concuerdan con la página.

Y así podemos nosotros hacer nuestro propio efecto:



Y de

manera dinámica podemos ver como como se va cambiando el fondo y los “pájaros” según los estilos que le apliquemos en la parte del menú lateral derecho.

- login.html:

```
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>UrbanVibe - Log in</title>
    <link rel="icon" href="img/UrbanVibeICON.ico" type="image/x-icon">
```

```

        <link rel="shortcut icon" href="img/UrbanVibeICON.ico" type="image/x-
icon">
    </head>
    <body>
        <div class="container">
            <div>

                <h1>UrbanVibe</h1>
                <br>
            </div>

            <form action="/login" method="POST">
                <div class="form-group">
                    <label for="username" class="form-
label">Username:</label>
                    <input
                        name="username"
                        id="username"
                        class="form-control"
                        type="text"
                        placeholder="user215"
                        required="required">
                </div>
                <div class="form-group">
                    <label for="password" class="form-
label">Password:</label>
                    <input
                        name="password"
                        type="password"
                        id="password"
                        placeholder="*****"
                        class="form-control"
                        required="required">
                </div>
                <div class="form-check">
                    <input class="form-check-input" type="checkbox"
id="rememberMe">
                    <label class="form-check-label" for="rememberMe">Recordar
sesión</label>
                </div>
                <div class="error-message" th:if="${param.error}">
                    Usuario y/o contraseña incorrectos
                </div>
                <a href="/users/signup" class="forgot-password">Don't you
have an account yet? Sign up!</a>
                <button type="submit" class="btn btn-primary">Sign
in</button>
            </form>
        </div>

```



```
</body>
</html>
```

Como podemos ver yo no uso un botón que me lleve al registro sino que uso una etiqueta “a” que me parece más cómodo para la experiencia del usuario. y luego en cuanto al registro:

- register.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>UrbanVibe - Sign up</title>
  <link rel="icon" href="img/UrbanVibeICON.ico" type="image/x-icon">
  <link rel="shortcut icon" href="img/UrbanVibeICON.ico" type="image/x-
icon">
</head>
<body>
  <div class="container">
    <div>
      <h1>UrbanVibe</h1>
      <br>
    </div>

    <form th:action="@{/users/signup}" th:object="${user}" method="POST">
      <div class="form-group">
        <label for="mail" class="form-label">Mail:</label>
        <input name="mail" type="email" id="mail" class="form-
control" placeholder="example@example.es" th:field="*{mail}" required>
      </div>
      <div class="form-group">
        <label for="username" class="form-label">Username:</label>
        <input name="username" id="username" class="form-control"
type="text" placeholder="user215" th:field="*{username}" required>
      </div>
      <div class="form-group">
        <label for="password" class="form-label">Password:</label>
        <input name="password" type="password" id="password"
placeholder="*****" class="form-control" required>
        <br/>
        <div th:if="${#fields.hasErrors('mail')}" th:errors="*{mail}"
class="error"></div>
        <div th:if="${#fields.hasErrors('username')}"
th:errors="*{username}" class="error"></div>

      </div>

      <div class="button-container">
```

```

        <button type="submit" class="btn btn-
primary">Register</button>
        <button type="submit" class="btn btn-primary"
onclick="window.location.href='/login'">Cancel</button>
    </div>
</form>
</div>
</body>
</html>

```

Y este es el registro y los estilos que he usado tanto en el registro como para el login son los siguientes:

```

body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    color: #333;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    padding: 0;
    background-image:
url('/img/Default_A_skyscraper_in_new_yorkalso_i_want_the_statue_of_libe_0.jpg');
    background-size: cover;
}

.container {
    width: 100%;
    max-width: 350px;
    padding: 50px;
    background-color: white;
    border-radius: 10px;
    box-shadow: 0px 4px 10px rgba(0,0,0,0.1);
    text-align: center;
}

h1 {
    color: black;
    margin-bottom: 10px;
    margin-top: 10px;
}

input.form-control {
    width: 100%;
    padding: 12px;
    margin-bottom: 20px;
    border: 1px solid black;
}

```

```

    border-radius: 5px;
    box-sizing: border-box;
    font-size: 16px;
}

label.form-label {
    color: black;
    font-weight: bold;
    display: flex;
    margin-bottom: 8px;
    text-align: left;
}

.button-container {
    display: flex;
    justify-content: space-between;
    margin-top: 20px;
}

.button-container button {
    flex: 1;
    margin-right: 10px;
}

button.btn-primary {
    background-color: black;
    border: none;
    color: white;
    padding: 14px;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease;
    font-size: 16px;
}

button.btn-primary:hover {
    background-color: black;
}

.error {
    color: red;
    font-size: 12px;
    margin-bottom: 10px;
}

```

Con estos estilos como podemos ver ajusto una imagen de fondo(Nueva York),con la intención de simular el lugar de venta principal de la tienda de ropa,además de por ejemplo mostrar los errores con color y de una forma muy visible como es el color rojo,he usado el flex:1 para que

ambos botones contengan el mismo ancho para el registro y algunos botones con sombra y sus respectivos hover.

## Bibliografía

Apunta aquí cada Web, tutorial, vídeo que veas y qué has aprendido con él:

- Tutorial para mandar correos de gmail: [https://www.youtube.com/watch?v=JKmzV1MY-M&t=2409s&ab\\_channel=UnProgramadorNace](https://www.youtube.com/watch?v=JKmzV1MY-M&t=2409s&ab_channel=UnProgramadorNace).
- Modificar login por defecto de Spring: <https://www.arquitecturajava.com/spring-boot-y-spring-security-custom-login/>
- Etiqueta Json en las clases modelo: <https://www.baeldung.com/jackson-bidirectional-relationships-and-infinite-recursion>
- Recursos web: <https://es.stackoverflow.com/questions/606452/login-y-error-con-java-spring-security-y-thymeleaf>  
<https://stackoverflow.com/questions/75534824/smtpsendfailedexception-and-mailconnectexception-while-sending-mail-using-java-m>
- Generador de imagenes: <https://leonardo.ai/>
- Creador de logo: <https://www.logomaker.com/>
- Outlook: <https://outlook.live.com/mail/0/>
- Documentacion <https://devdocs.io/html/>
- Vanta.js <https://www.vantajs.com/>

## Proyecto de Desarrollo de Software

Podemos generar un libro en formato imprenta con este comando:

```
docker run --rm \
  --volume "$(pwd):/data" \
  --user $(id -u):$(id -g) \
  pandoc/extra 0-*.md -o \
  Libro.pdf --template eisvogel --listings --number-sections
```

Para generar la presentación en HTML (desde la carpeta diapositivas):

```
docker run --rm -v $PWD:/home/marp/app/ -e LANG=$LANG marpteam/marp-cli
presentacion.md
```

## Extensiones de VS Code

- Marp for VS Code
- Git Graph
- Markdown All in One

- Markdown Preview Enhanced
- Markdown Shortcuts
- Java Pack
- Spring-boot
- Docker
- Database(Connector)
- Beauty(Dar formato al código)
- Error Lens(Indica los errores en el código)
- Gitlab Workflow
- IntelliJCode
- Tabnine(ID)
- Markdownlint
- Dracula(UI de Vscodex)
- Autoclose tag(HTML)