UNIVERSIDAD DE GUANAJUATO

DIVISION DE INGENIERIAS

COMPUTO EN LA NUBE

PROYECTO “WE SPORT”

JUAN ADOLFO GUTIERREZ GAYTAN

Contents

INTRODUCCION

La región sur de Guanajuato es conocida a nivel nacional por su gran producción textil, a tal grado de que muchas personas viajan desde distintos puntos del país a los municipios de Uriangato y Moroleón solo para comprar productos tales como: playeras, pantalones, calcetines, ropa interior, etc.

El proyecto “We Sport” se enfoca dentro de todos estos productos al apartado de playeras conocidas como “Playeras tipo FOX”, desde que se hace el pedido hasta que se venden.

Veremos como la metodología XP es empleada desde el inicio de un proyecto hasta el final de uno, como entran en juego la fase de planificación, diseño, codificación, pruebas y lanzamiento, y como gracias a la unión de las tecnologías de Docker, Git, GitHub, JavaScrip y JSON se puede desarrollar una aplicación desde 0.

PROBLEMÁTICA.

La empresa We Sport se dedica a la producción de Playeras tipo FOX, para entender un poco el contexto de la problemática debemos conocer el proceso de producción de una playera:

Paso 1:

Se crea el diseño de la playera, este diseño puede ser original o ya existente en internet.

Imagen 1: Playera FOX

Paso 2:

Se manda a impresión, esta acción se lleva a cabo en una maquina conocida como Plotter.

Imagen 2: Plotter

Paso 3:

Después de la impresión el papel de manda a planchado, donde se transfiere la imagen a la tela.

Imagen 3: Sublimado.

Sabiendo esto, comenzaremos con la problemática, en esta empresa cualquier persona puede realizar un pedido con las playeras que el quiera, a estos pedidos se les llaman bitácoras, las bitácoras se realizan en Excel por la encargada de ventas, esta bitácora es pasada al encargado de impresión en forma de imagen, mensaje de texto o hoja de papel impresa, al hacer esto no se lleva un control efectivo de las bitácoras así como de quien realiza cada bitácora, esto lleva a la problemática de que los pedidos no se entreguen a tiempo o no coincidan las cantidades.

GHYSLAINE 24/11/2021				PEDIDO D22		
MODELO	DESCRIPCIÓN	TALLA	CANTIDAD	JUMA	TIENDA	TOTAL
PLIZADO	PODER JUDICIAL AZUL	ADULTO	35			
PLIZADO	PODER JUDICIAL NEGRO	ADULTO	35			
PLIZADO	PODER JUDICIAL GRIS	ADULTO	35			
A78	PLAYERA DEPORTIVA FOX	MD	1			
A78	PLAYERA DEPORTIVA FOX INFALTIL	GD	1			

Imagen 4: Bitácora de impresión.

Después de que el encargado de impresión recibe la bitácora, existe un problema, si tenemos en cuenta experiencia de la persona que está a cargo de la impresión, al momento de ver los códigos de los modelos puede que no reconozca o no relacione la imagen de la playera con los códigos, existe un catálogo en formato PDF, pero este no se actualiza de forma constante o conforme salen nuevos diseños.

Otra problemática es que no se lleva un control efectivo de quien realiza las ventas y esas ventas no se ven reflejadas inmediatamente en el inventario que existe en la tienda.

DESARROLLO.

Para el desarrollo de la aplicación web se aplicó la metodología XP, ya que esta metodología es la que mas se acopla al proyecto, esta metodología nos permite una gran comunicación entre el programador y el cliente, podemos realizar pruebas continuas a lo largo del desarrollo, puede ser aplicada a cualquier lenguaje de programación.

PLANIFICACIÓN:

Comenzamos con la etapa de planificación, recaudamos las historias de usuario provenientes del cliente, los cuales fueron:

1. Como usuario de la aplicación quiero una pantalla una pantalla de registro de usuarios para poder crear una cuenta.
2. Como usuario de la aplicación quiero una pantalla de inicio de sesión para poder entrar a la aplicación con la cuenta que cree.
3. Como usuario de la aplicación quiero que esta tenga una contraseña encriptada para mayor seguridad.
4. Como usuario de la aplicación quiero que esta tenga diferenciación de usuarios para que cada cuenta tenga sus propios registros.
5. Como usuario de la aplicación que tenga una pantalla donde se muestre el catálogo de las playeras para así poder ver los elementos.
6. Como usuario de la aplicación quiero poder subir, editar y eliminar elementos del catálogo.
7. Como usuario de la aplicación quiero que la pantalla del catálogo sea común para todas las cuentas para que cada usuario no tenga que hacer el propio.
8. Como usuario de la aplicación quiero una pantalla del inventario para poder ver lo que tengo en existencia.
9. Como usuario de la aplicación quiero poder subir, editar, eliminar y vender elementos desde la pantalla de ventas.
10. Como usuario de la aplicación quiero una pantalla para registrar bitácoras.
11. Como usuario de la aplicación quiero poder crear, editar y eliminar las bitácoras.
12. Como usuario de la aplicación quiero poder agregar playeras a esas bitácoras.
13. Como usuario de la aplicación quiero poder eliminar y editar esas playeras.
14. Como usuario de la aplicación quiero poder ver las bitácoras completas.
15. Como usuario de la aplicación quiero que las bitácoras sean únicamente de la cuenta que las cree para evitar confusión entre cuentas.
16. Como usuario de la aplicación quiero una pantalla donde se registren las ventas automáticamente.
17. Como usuario de la aplicación quiero poder eliminar los registros de ventas.
18. Como usuario de la aplicación quiero que los registros de venta sean únicamente de la cuenta que las cree para poder llevar un registro de que usuario vende cada cosa.

19. Como usuario de la aplicación quiero una pantalla para poder ver los datos de la cuenta en uso.
20. Como usuario de la aplicación quiero poder editar la información del usuario, poder eliminar la cuenta y al eliminar la cuenta se elimine todo lo que haya hecho ese usuario.

Teniendo en cuenta las historias del usuario concluimos que la aplicación contara con los siguientes apartados:

1. Apartado de Sesión.
2. Catalogo.
3. Inventario.
4. Registros de ventas.
5. Perfil.

Basándonos en las historias de usuario definidas para el desarrollo de la aplicación se desarrollo el siguiente plan de entrega:

No de historia	iteración	Prioridad	Fecha de inicio	Fecha Final
1	1	Alta	29/11/2021	29/11/2021
2	1	Alta	29/11/2021	29/11/2021
3	1	Alta	29/11/2021	29/11/2021
4	1	Alta	29/11/2021	29/11/2021
5	1	Alta	30/11/2021	30/11/2021
6	2	Alta	30/11/2021	30/11/2021
7	2	Alta	30/11/2021	30/11/2021
8	2	Alta	30/11/2021	30/11/2021
9	2	Alta	01/12/2021	01/12/2021
10	2	Alta	01/12/2021	01/12/2021
11	2	Alta	02/12/2021	02/12/2021
12	2	Alta	02/12/2021	02/12/2021
13	3	Alta	03/12/2021	03/12/2021
14	3	Alta	03/12/2021	03/12/2021
15	3	Alta	04/12/2021	04/12/2021
16	3	Alta	04/12/2021	04/12/2021
17	3	Alta	05/12/2021	05/12/2021
18	3	Alta	05/12/2021	05/12/2021
19	3	Alta	06/12/2021	06/12/2021
20	3	Alta	06/12/2021	06/12/2021

Tabla 1: Prioridades de historias

DISEÑO.

Metáfora del sistema:

El sistema We Sports será capaz de diferenciar entre usuarios, se podrá agregar, eliminar y editar elementos del catálogo, de igual forma se podrá agregar, eliminar, editar y vender elementos del inventario, podremos visualizar las ventas realizadas, podremos crear, editas y eliminar bitácoras, por último, tendremos un apartado para visualizar, editar y eliminar las cuentas.

Mockup:

Teniendo en cuenta los requisitos del cliente comenzamos a realizar unos mockups muy simples para que sirvieran de guía a la hora de programar.

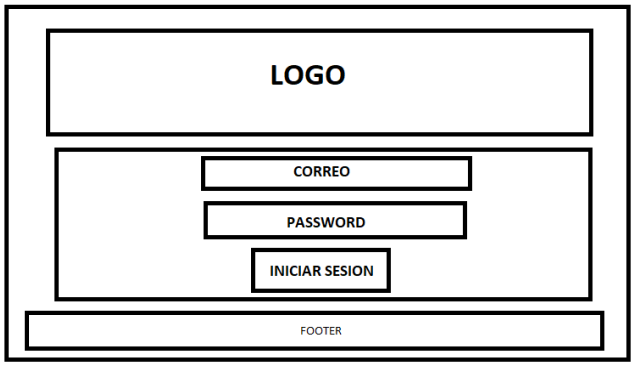


Imagen 5: Mockup Login

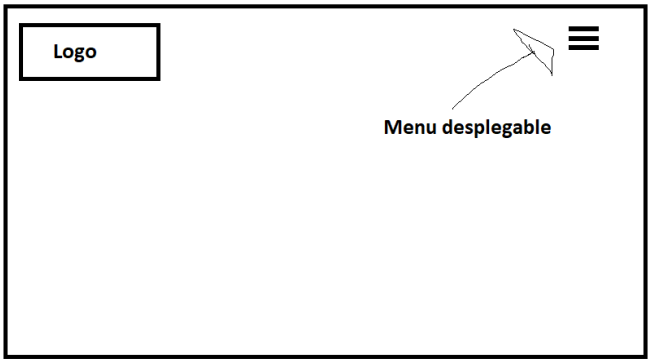


Imagen 6: Mockup menú

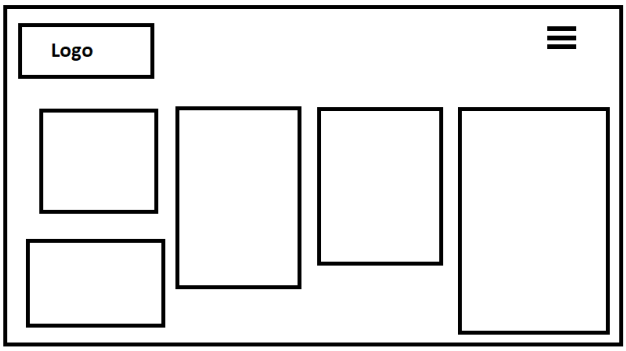


Imagen 7: Mockup catálogo, inventario y ventas

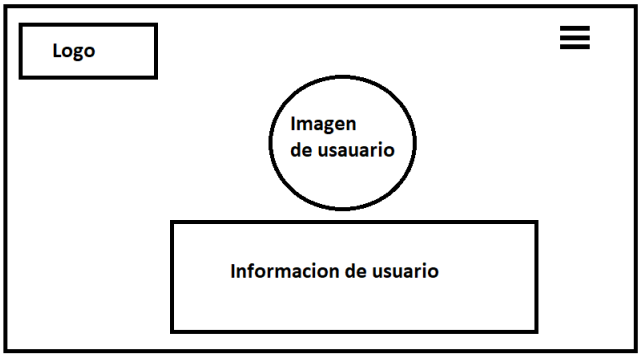


Imagen 8: Mockup perfil

Tarjetas CRC:

Seguido de esto en esta etapa se realizaron las tarjetas CRC o también conocidas como Class-responsibility-collaboration card, estas tarjetas son una herramienta de lluvia de ideas las cuales ayudan a la planificación de los atributos que tendrán las clases.

Nombre: Usuario	**id:** usuario	**Tipo:** publico
Descripcion: en esta clase se lleva el control de la informacion del usuario	**Casos de uso asociados:**	

Tabla 1: CRC usuario.

Nombre: Inventario	**id:** inventario	**Tipo:** publico
Descripcion: En esta clase se lleva el control de la informacion del inventario	**Casos de uso asociados:** Usuario	

Tabla 2: CRC inventario.

Nombre: Bitacora	**id:** bitacora	**Tipo:** publico
Descripcion: En esta clase se lleva el control de la informacion de la bitacora	**Casos de uso asociados:** Usuario	

Tabla 3: CRC bitácora.

Nombre: Playera	**id:** playera	**Tipo:** publico
Descripcion: En esta clase se lleva el control de la informacion de la playera	**Casos de uso asociados:** Usuario	

Tabla 4: CRC playera.

Nombre: Venta	**id:** venta	**Tipo:** publico
Descripcion: En esta clase se lleva el control de la informacion de la venta	**Casos de uso asociados:** Usuario	

Tabla 5: CRC venta.

CODIFICACIÓN.

Estándares de codificación:

Se definió como primer paso los estándares a seguir para la codificación los cuales fueron:

- **Generales.**
  - El código de este sistema debe de estar desarrollado con el lenguaje de etiquetas HTML usando el sistema de plantillas de Express.
  - Usar Bootstrap como framework para frontend.
  - Usar Docker para el entorno de programación.
  - Utilizar JSON para construir el servidor.
- **Nombres de variables.**
  - Los nombres que se usen deben ser significativos.
  - Los nombres deben estar en minúsculas, excepto la primera letra de cada palabra a partir de la segunda.
- **Indentación.**
  - Todo código desarrollado tendra una indentacion de una tabulacion, esto para identificar mas facilmente las partes del codigo. Ejemplo:

```
const storage = multer.diskStorage({
  destination: path.join(__dirname, "public/img/up"),
    filename: (req, file, cb, filename) => {
      console.log(file);
      cb(null, uuidv4() + path.extname(file.originalname));
    },
});
```

Figura 1: Indentación.

- **Comentarios.**
  - Todas las funciones tienen que estar documentadas explicando que realiza cada una de ellas.
- **Espacios dentro del código.**
  - Se deben dejar espacios de separación entre los elementos para poder identificar con mayor facilidad los elementos
- **Entrega de avances.**
  - Cada que se complete una historia de usuario se debe de subir el cambio a la plataforma de GitHub, este deberá de contener quien hizo el cambio y también un comentario con que fue lo que se modificó o se agregó.

Integración del código:

Para el apartado de control de versiones y entrega de código se utilizó la herramienta Git y GitHub, en donde se subían los cambios de código.

Imagen 9: Commits en GitHub

Configuración del proyecto:

Seguido de esto se eligió la tecnología a utilizar, la cual fue:

- Docker.
- MongoDB.
- JS.
- JSON.
- Bootstrap.

Se utilizo Docker como herramienta principal donde estaremos "instalando" nuestras herramientas, en el instalamos las imágenes de MongoDB, Node y Mogo Express. MogonDB será nuestro motor de base de datos, Node nos sirve para montar nuestro servidor sin necesidad de algún programa externo y Mogo Express será nuestro ID de MongoDB.

Para poner en marcha nuestro servidor y nuestra aplicación necesitamos instalar y configurar algunos componentes:

- Bcrypt-nodejs: Este módulo sirve para encriptar las contraseñas.
- Connect-flash: es un paquete para Express que nos permite mostrar mensajes en la pantalla bajo ciertas condiciones.
- Ejs-mate: sirve para importar fragmentos de código común.
- Express: Proporciona mecanismos para escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL, integración con motores de renderización de vistas para generar respuestas mediante la introducción de datos en plantillas
- Express-session: es un middleware que almacena los datos de sesión en el servidor.
- Fs-extra: es una extensión del módulo fs del sistema, que proporciona API mas convenientes y hereda la API del módulo fs.
- Mongoose: es una librería para Node.js que nos permite escribir consultas para una base de datos de MongoDB.

- Multer: es un middleware para Express y Node.js que hace más fácil manipular este tipo de inputs: multipart/form-data cuando los usuarios suben archivos.
- Passport: framework para construir la autenticación de la aplicación de Node.js
- Uuid: es un identificador único.
- Nodemon: es una utilidad que monitorea los cambios en el código fuente que se está desarrollando y automáticamente reinicia el servidor.

Seguido de eso pasamos a configurar nuestros documentos referentes a Docker:

- Dockerfile.
- docker-compose: donde instalaremos las imágenes y en el contenedor donde se van a instalar.

El framework a utilizar para darle una apariencia más estética fue Bootstrap el cual es un framework de interfaz de usuario, de código abierto, creado para un desarrollo web más rápido y sencillo. Contiene todo tipo de plantillas de diseño basadas en HTML y CSS para diversas funciones y componentes, como navegación, sistema de cuadrícula, carruseles de imágenes y botones.

Después de definir tecnologías a utilizar pasamos a definir la ramificación de las carpetas a utilizar.

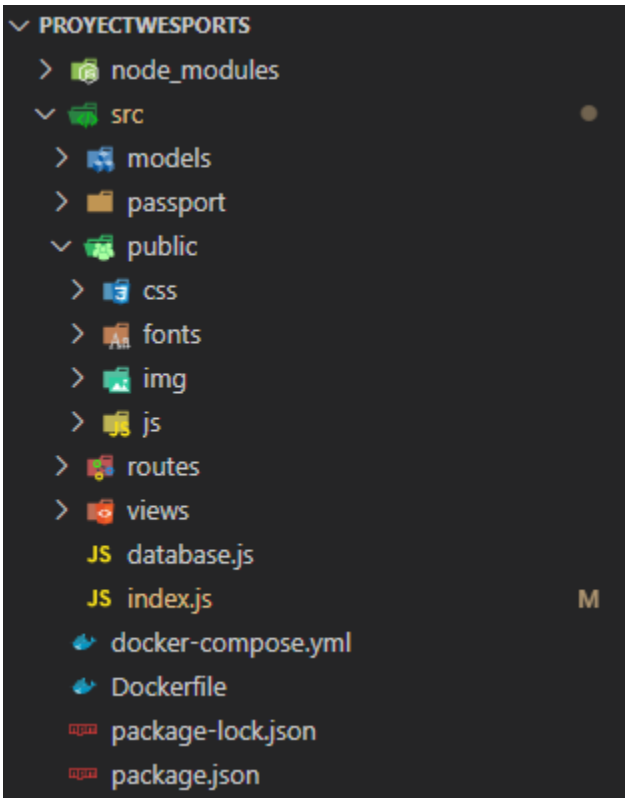


Imagen 10: Carpetas.

Pasaremos a describir cada capeta de nuestro proyecto:

- node_modules: carpeta creada al momento de iniciar nodejs es necesaria para su funcionamiento.
- Models: carpeta donde se almacenan los modelos que se utilizan para crear las colecciones de mongodb.
- Passport: aquí se almacena el método para realizar la autenticación de usuarios.
- Public: esta carpeta se subdivide en 4 más:
    - Css: los estilos utilizados para las vistas.
    - Fonts: carpeta para las fuentes.
    - Img: aquí se guardan las imágenes necesarias.
    - Js: archivos que se utilizan para las animaciones de algunos elementos.
- Routes: archivo que contienen las rutas a las cuales el servido accede y redirige a la vista solicitada.
- Views: Todas las pestañas que son necesarias para la aplicación.

Creación de modelos:

Como sabemos MongoDB utiliza modelos para poder crear lo que en otros motores de base de datos se les conoce como tablas, para este proyecto se necesitaron crear 6 modelos:

- Image (Catalogo).
- Inventario.
- Product (Bitacora).
- Tshirt.
- Users.
- Venta.

Image	
Title	String
Description	String
Filename	String
Path	String
Originalaname	String
Mimetype	String
Size	Number
Created	Date

Inventario	
title	String
Description	String
Cantidad	Number

Product	
Nombre	String
Cliente	String
Pedido	Number
Fecha	String
Usuario	String

Tshirt	
modelo	String
Descripción	String
Talla	String
Cantidad	String
Usuario	String
Bitacora	String

Users	
Name	String
Puesto	String
Email	String
Password	String

Venta	
cantidadV	Number
Modelo	String
Fecha	String
Usuario	String

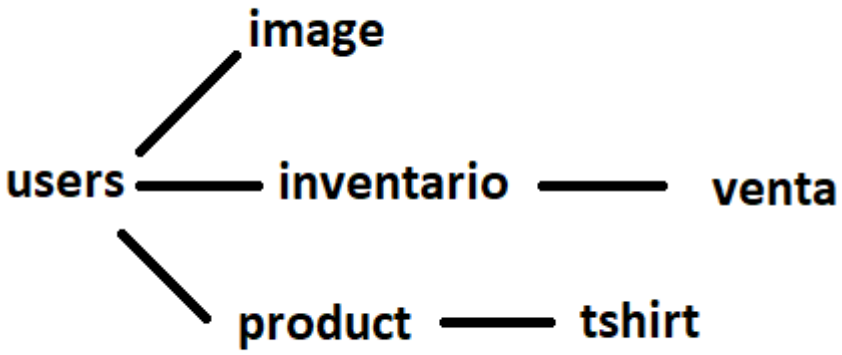


Imagen 11.-Relacion de Modelos.

Creación de rutas.

Como sabemos nuestro proyecto funciona mediante archivos enlazados entre sí, pero debemos indicarle a nuestro servidor que hacer cuando se le hace una petición con dichas rutas, para ello necesitamos configurarlas, para este proyecto se decidió definirlas de la siguiente manera:

```
router.get('/',(req,res, next)=>{
    res.render('index');
});
```

Figura 2.-Ruta

Como se observa lo que hacemos es que cuando el servidor recibe una petición con la dirección de nuestro servidor (http://localhost:3000/) lo que hare el servidor es renderizar o mostrar la vista index.

Esto se realizó con cada una de las vistas que se tienen en el proyecto.

Elementos de Bootstrap utilizados.

- Botones:

  Se utilizaron los botones que proporciona Bootstrap para las acciones a relizar

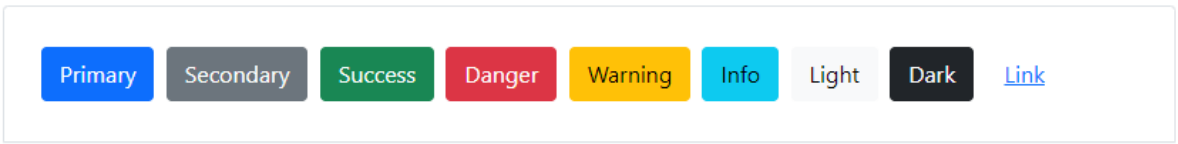


Imagen 12.-Botones

- Tarjetas:

  Las tarjetas se utilizaron para mostrar información en la pantalla de una forma mas dinámica.

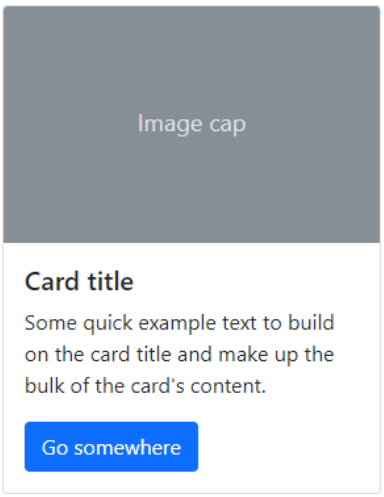


Imagen 13.-Tarjeta

- Formulario:

  Los formularios se utilizaron para ingresar datos a la base de datos.

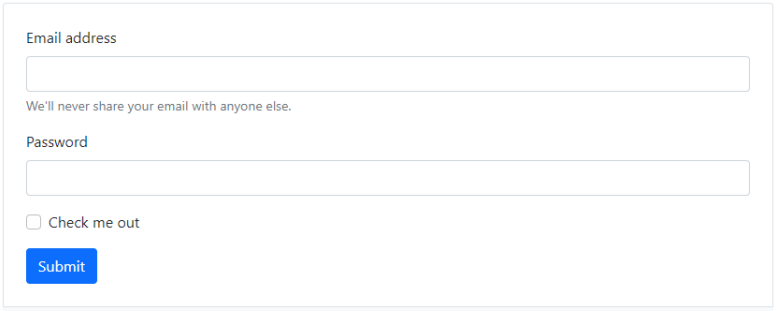
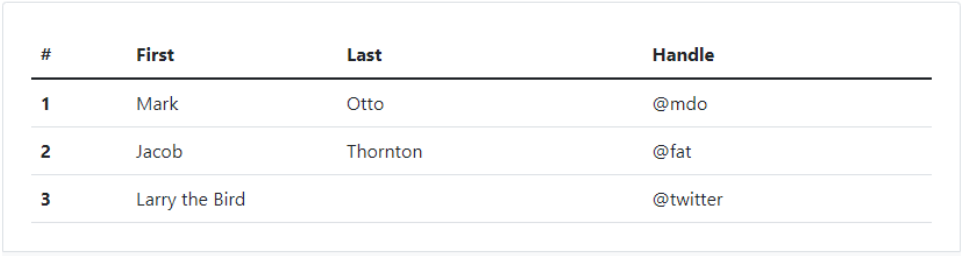


Imagen 14.-Formulario

- Tabla:

  Las tablas fueron requeridas para mostrar la información de las bitácoras.



#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

Imagen 15.-Tabla

- Iconos:

  Los iconos se utilizaron en los botones para hacerlos más intuitivos.

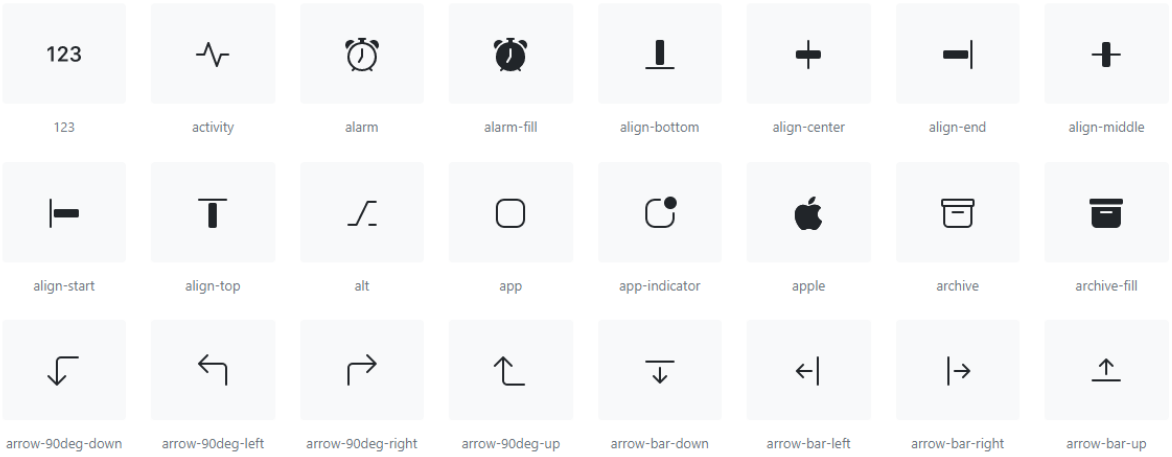


Imagen 16-Iconos

PRUEBAS.

Las pruebas fueron hechas manualmente ya que por el tiempo estimado de realización resultaba más rápido.

Las pruebas más tardadas fueron las validaciones, como sabemos esta es una de las partes más importantes, ya que sin esto la información de nuestro sistema se ve comprometida, y ya no es fiable.

La parte del código a probar es el siguiente:

```
if(titulo.length == 0){
    req.flash('signupMessage', 'Introduce un Codigo');
    res.redirect('agregarinventario');
  }else if(titulo.charCodeAt(0) == 32){
    req.flash('signupMessage', 'Introduce un Codigo');
    res.redirect('agregarinventario');
  }else if(des.length == 0){
    req.flash('signupMessage', 'Ingresa una descripcion');
    res.redirect('agregarinventario');
  }else if(des.charCodeAt(0) == 32){
    req.flash('signupMessage', 'Ingresa una descripcion');
    res.redirect('agregarinventario');
  }else if(can.length == 0){
    req.flash('signupMessage', 'Ingresa una cantidad');
    res.redirect('agregarinventario');
  }else if(can.charCodeAt(0) == 32){
    req.flash('signupMessage', 'Ingresa una cantidad');
    res.redirect('agregarinventario');
  }else if(can < 0){
    req.flash('signupMessage', 'Ingresa una cantidad correcta');
    res.redirect('agregarinventario');
  }else{
    inventario.title = req.body.title;
    inventario.description = req.body.description;
    inventario.cantidad = req.body.cantidad;
    inventario.path = '/img/up/' + req.file.filename;
    await inventario.save();
    req.flash('signupMessage', 'Producto Guardado');
    res.redirect('inventario');
  }
```

Figura 3: código de validación

Como podemos observar en la figura 2, el código valida los campos de texto, compara mediante comparaciones si las cadenas que llegan están vacías o solo son espacios.

Para comprobar que el código funciona se realizaron inserciones desde la aplicación hasta que se logo la funcionalidad al 100% de estas validaciones.

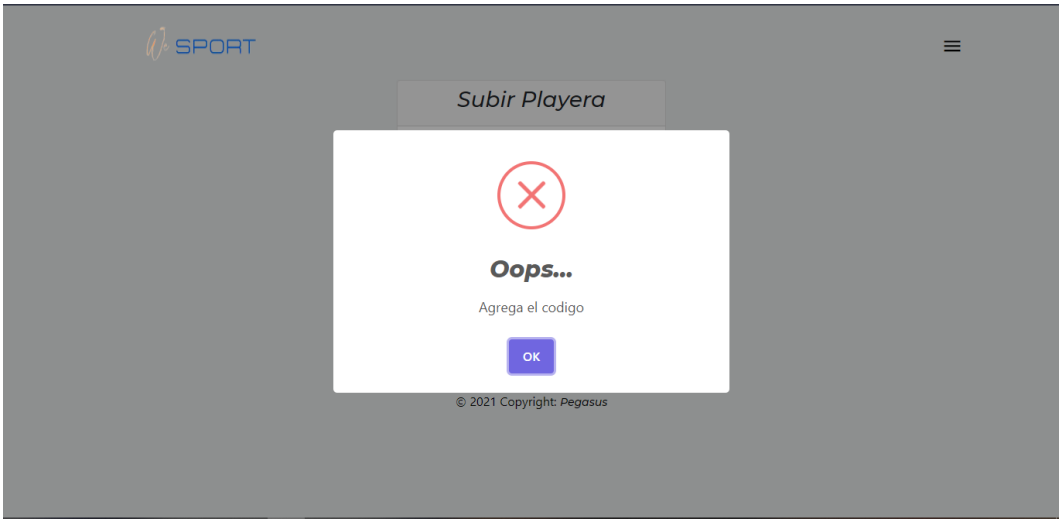


Imagen 17: prueba.

De igual forma se realizaron pruebas en todos los demás apartados del sistema, por ejemplo, la pantalla de registro de un nuevo usuario, el sistema no debe de permitir dos usuarios con el mismo correo, ya que esto llevaría a un error muy grave en el sistema.

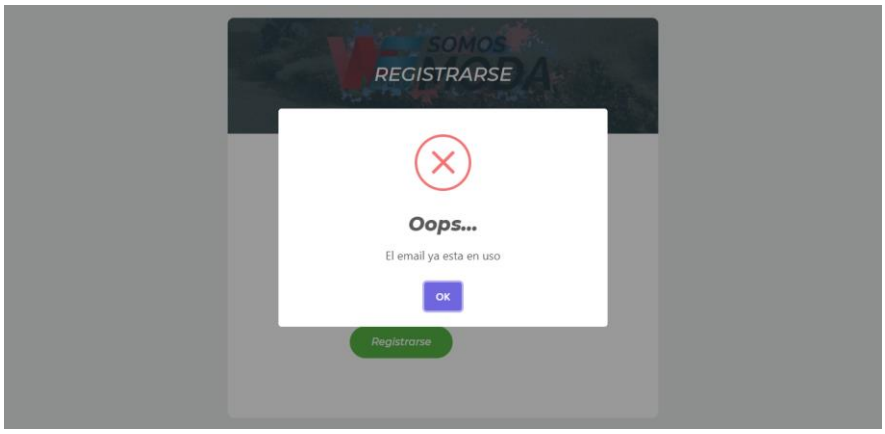


Imagen 18: validación de usuario.

Otras de las pruebas clave al crear un proyecto y que este maneje una base de datos es la comprobación de que la aplicación este consumiendo la base de datos.

```
mongoose.connect('mongodb://mongo/WeSports')
.then(db => console.log('Base conectada',db.connection.host))
.catch(err => console.error('ERROR',err))
```

Figura 4: conexión a la base de datos.

En este método nos conectamos a la base de datos, como podemos ver la comprobación de que si la conexión es exitosa es mediante la consola, ya que nos enviara un mensaje como el siguiente:

Imagen 19: conexión a base de datos.

Como se observa la conexión es exitosa.

CONCLUSION.

Podemos concluir que gracias a la implantación de la tecnología de Docker y Github se puede trabajar de una maneja mucho mas eficiente y rápida, ya que a lo largo del desarrollo de este proyecto se estuvo trabajando en dos computadoras y el uso de estas tecnologías hacía mucho más rápido la migración e implementación de los cambios realizados en una computadora.

En cuanto a la metodología utilizada es de gran ayuda ya que la metodología XP está hecha para proyectos a corto plazo, y esto hace mucho más eficiente el desarrollo, de tal forma que si existe algún error en la codificación podemos regresar a la planeación de forma inmediata, también es de gran ayuda que al entregar los avances se realicen las pruebas esto para que al final no se acumulen todas.

Como resultado final obtuvimos una aplicación la cual cumple con todas características que el cliente solicito y resuelve la problemática.

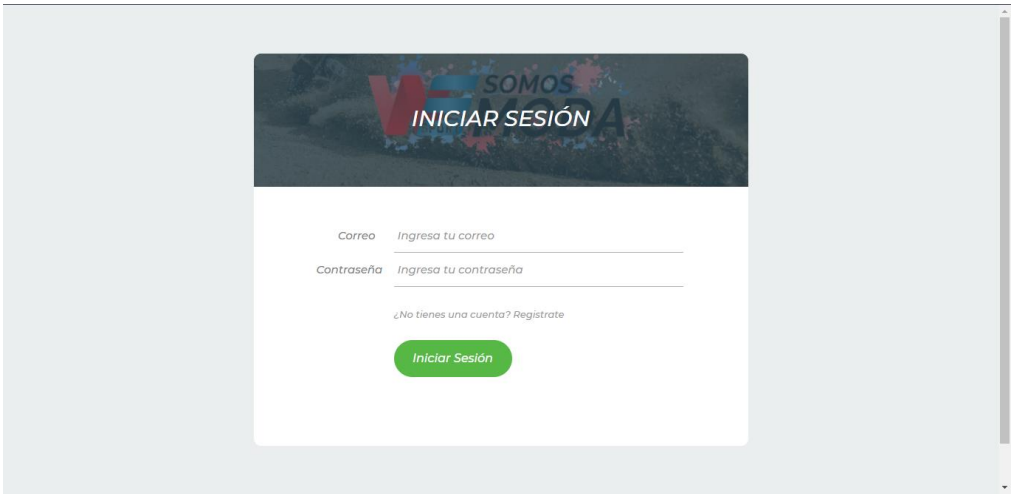


Imagen 20: login.

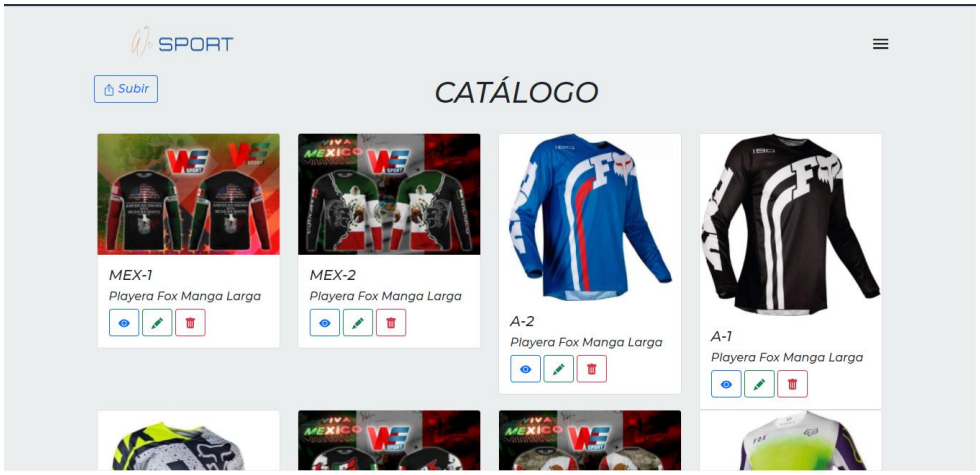


Imagen 21: Catalogo.

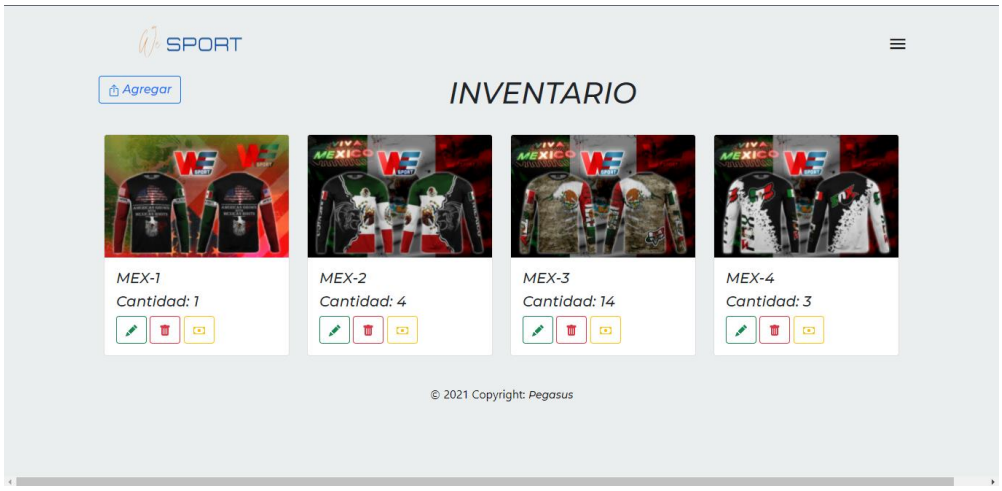


Imagen 22: Inventario.

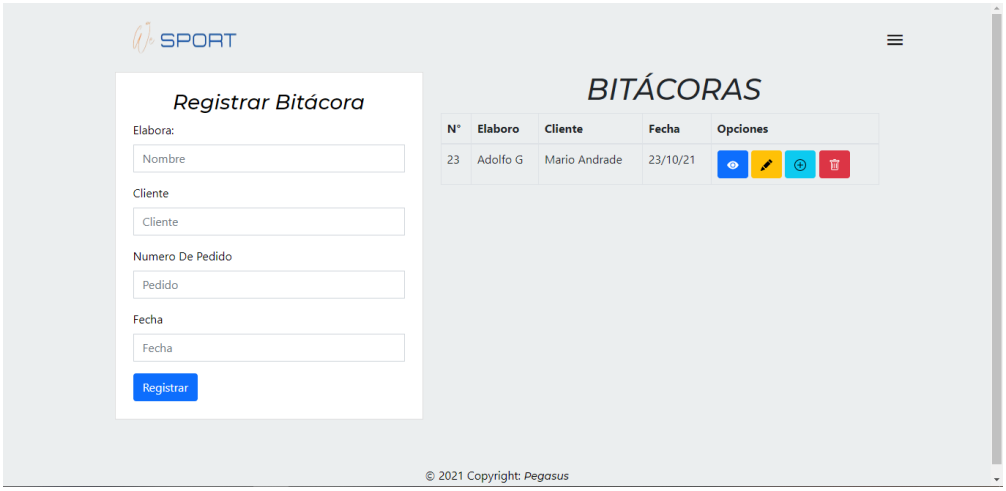


Imagen 23: Bitácoras.

Imagen 24: Ventas.

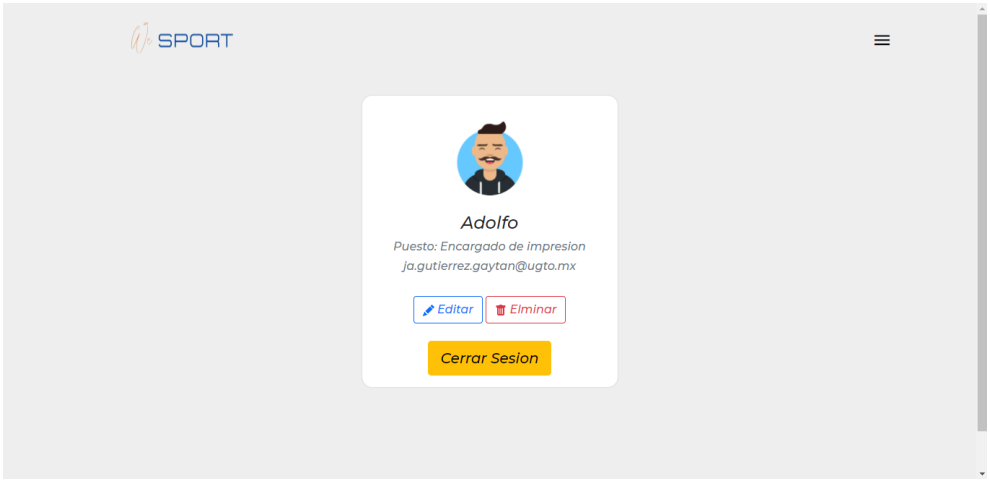


Imagen 25: Perfil.

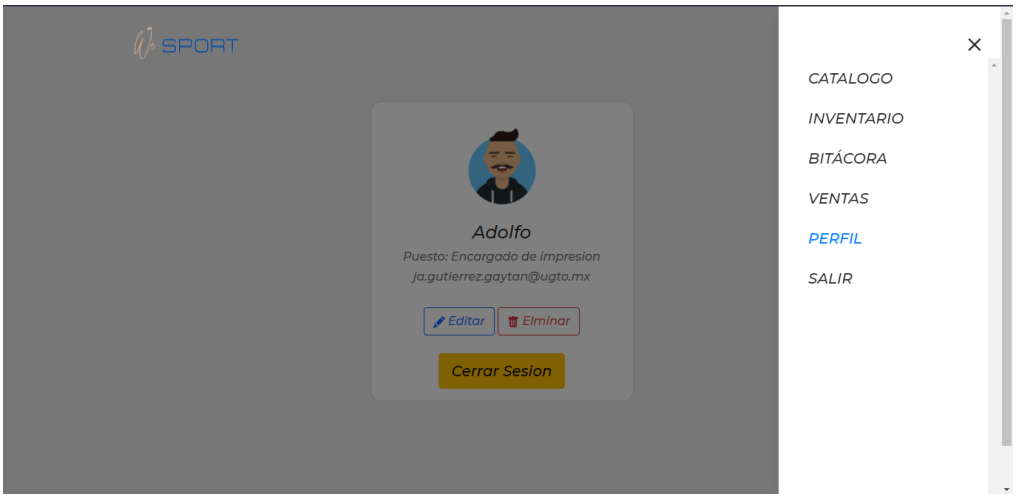


Imagen 26. Menú