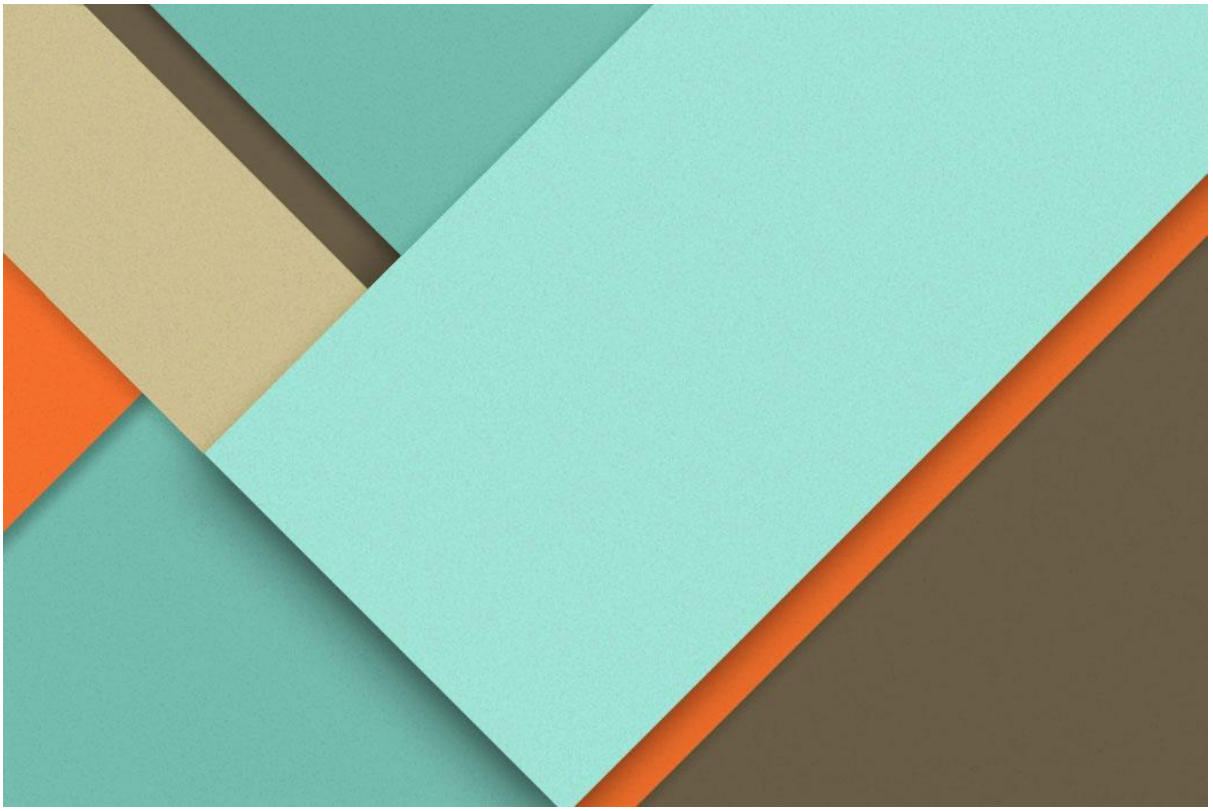


Book S.A

13/06/2021

Versión 2.0

Grupo 18

Software Avanzado.

Escuela de ciencias y sistemas.

Universidad de San Carlos de Guatemala.

- Resumen de la aplicación a desarrollar

El proyecto de "*BookSA*" surge por la solicitud de la empresa para atender a todos aquellos clientes que estén dispuestos a comprar libros online y asociado a ello reducir costos tanto de operación, arrendamiento de local así como mejorar el servicio y llegar a un público más amplio que antes cuando solo constaba de locales físicos. De manera general se trata de la realización de una tienda en línea a nivel nacional y con miras a una posible expansión internacional que está enfocada en la venta de libros por parte de diferentes editoriales actuando de intermediario entre los mismos y los clientes finales agilizando el proceso. El proceso de desarrollo consta de 3 fases las cuales se detallan a continuación:

- Fase 1: Esta fase se enfoca en crear los servicios para registrar usuarios y editoriales, habilitar las funciones básicas del administrador del sitio y la aceptación de editoriales nuevas por parte del mismo. Las editoriales podrán gestionar sus bibliotecas creando libros, modificandolos y eliminándolos según se requiera. A su vez los usuarios podrán comprar libros de diferentes editoriales y realizar las compras seleccionando distintas formas de pago.

- Fase 2: Consta de lograr escalar de forma horizontal la aplicación resultante de la fase 1, creación de formularios para que un cliente solicite un libro que no exista, creación de una bitácora para registrar los movimientos que realice una editorial y, por solicitud de varias de las editoriales una función de cálculo de impuesto para varios países dependiendo de donde deseen vender los libros. Las nuevas funcionalidades serán implementadas con una arquitectura de microservicios.

- Fase 3: Se plantea agregar nuevas funcionalidades al sitio de "BookSA", este contará con un manejo de órdenes realizadas con 5 tipos de estados según el progreso del envío, también se agregará un servicio de entrega de facturas al correo electrónico. Por último se integrará al sitio web un Servicio de Bus Empresarial (ESB) que está enfocado en la arquitectura de servicios y el cual gestionará los servicios de signup, login, visualización de productos y compras.

Funcionalidades de la Fase 1

- App Web (responsiva y en tiempo real)
- Tipos de usuarios:
    - Administrador (creado desde el despliegue de la aplicación)
    - Editorial
    - Clientes
- Login (todos los usuarios)
- Registro (solo editorial y cliente)

Datos a registrar:

- Editorial (esperar aprobación del administrador)
    - Nombre
    - Correo
    - Contraseña
    - Dirección Física
- Cliente
    - Nombres
    - Apellido
    - Correo
    - Contraseña
    - Número celular

Funcionalidades:

- Administradores
    - Aceptar editoriales
    - Eliminar usuarios (tipo editorial o tipo cliente)
- Editorial
    - Crear, ver, actualizar y eliminar libros (estos pueden pertenecer a uno o más géneros literarios).
    - Actualizar el stock de sus productos (libros con stock de 0 no aparecen en el catálogo para los clientes).
- Cliente
    - Navegar por el catálogo de libros
    - Gestionar el carro de compras
    - Realizar el pago seleccionando un método de pago y método de envío.

Funcionalidades de la Fase 2

- Escalamiento horizontal para los servidores de peticiones debido a la alta demanda del sistema.

Datos a registrar:

- Bitácora:
    - Tipo de actividad
    - Nombre de editorial
    - Fecha
- Libros inexistentes:
    - Nombre del libro
    - Nombre del autor
    - Fecha de publicación
    - PDF (opcional)

Funcionalidades:

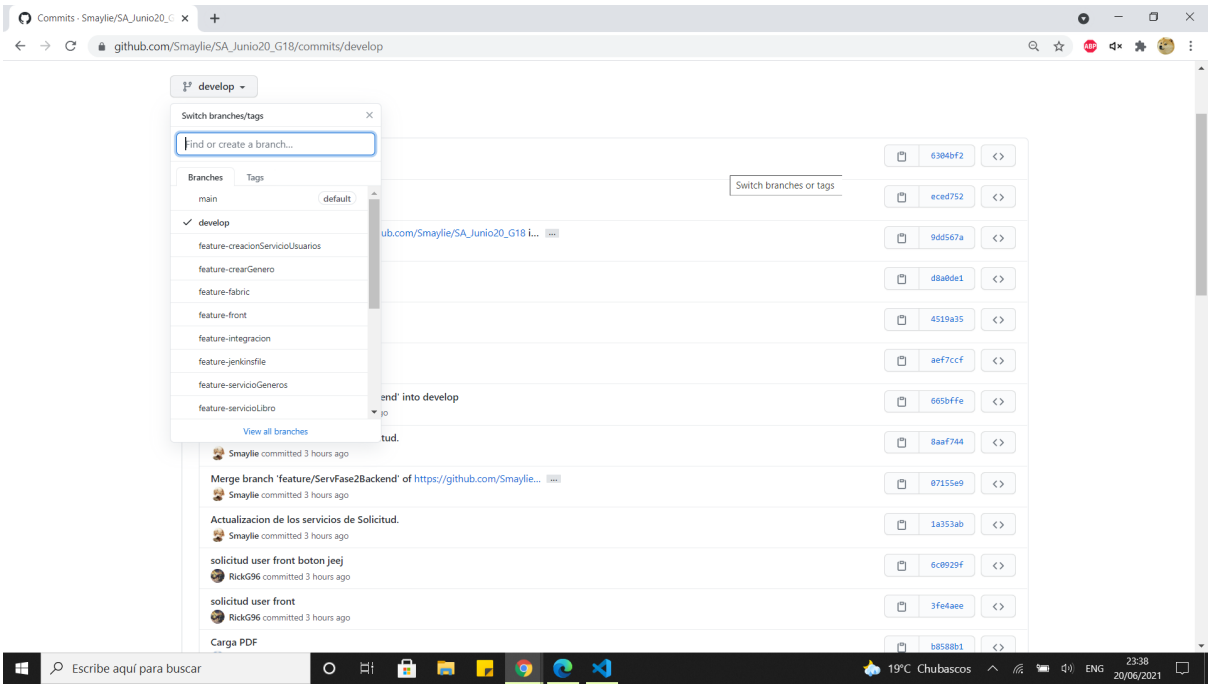
- Administrador:
    - Monitoreo de control de actividades de las editoriales.
- Usuarios:
    - Solicitar libros inexistentes.
- Editoriales:
    - Nuevo portal para cálculo de impuestos.
    - Aceptar solicitudes de libros

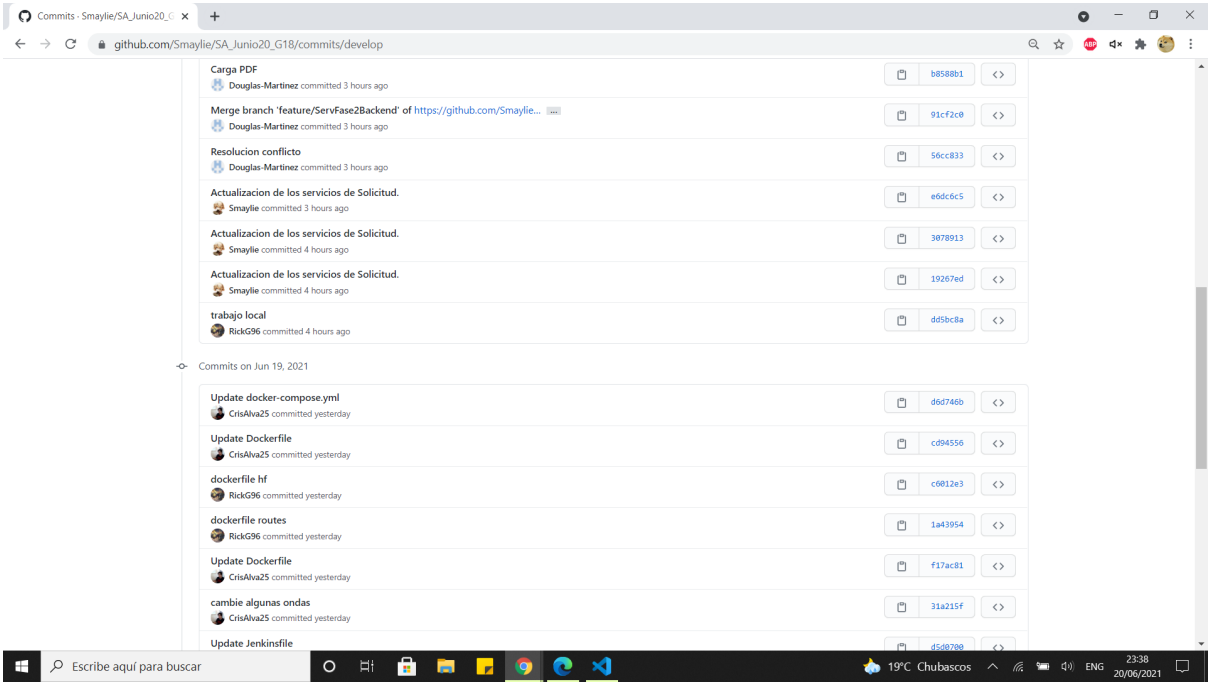
- Versionamiento del documento

Para el versionamiento del producto se utilizará un repositorio privado de *GitHub* que utiliza como base el software de control de versiones *Git*. En cuanto a la estrategia del flujo de trabajo se utiliza *git flow* ya que se acopla muy bien al desarrollo de proyectos con entregas iterativas como es el caso de “Book S.A.”.

- Link del repositorio: https://github.com/Smaylie/SA_Junio20_G18

A continuación se detalla el estado del producto, forma del versionamiento, uso de git flow y contribución de los integrantes del equipo:





- Lenguaje de programación

Al ser solicitado una aplicación de sitio web se consideraron para el desarrollo del mismo las tecnologías más documentadas, detalladas, de código abierto y robustas para el desarrollo web y como resultado de dicho análisis se eligieron aquellas que se utilizarán frameworks que tienen a ***JavaScript*** y ***Typescript*** como lenguaje de programación base.

- Consideraciones: a pesar de que *SQL* no es un lenguaje de programación si se utilizará como lenguaje de consultas ya que la base de datos a implementar es de tipo relacional.

- Herramientas de desarrollo a utilizar

Para realizar la parte frontal del sitio (diseño, vistas, colores, tamaños, etc) se utilizará ***Angular*** en su versión 10.1.4 que utiliza Typescript como lenguaje de programación y se apoya de otros lenguajes como Javascript, HTML, CSS y SCSS. La elección se basa en que es una herramienta que permite realizar aplicaciones “single-page” de manera sencilla además de que es multiplataforma lo cual hace que nuestra aplicación sea portable.

En cuanto a la capa de servidor se utilizará el entorno de ejecución ***Node.js*** el cual utiliza JavaScript como lenguaje de programación. Node.js también es multiplataforma asegurando la portabilidad, se maneja de forma asíncrona lo cual

hace más eficiente y fluida la experiencia del usuario y por último se consideró que Node.js se orienta a eventos lo cual es ideal para el acoplamiento con el el frontend e interacción con las actividades del usuario. Para hacer más fácil la creación del servidor se utilizará el framework de infraestructura de aplicaciones web de Node.js llamada ***ExpressJs*** ya que proporciona una capa de características web y móviles básicas predefinidas además de que facilita la creación de API ya que tiene varios métodos HTTP y de middleware disponibles.

En la parte de almacenamiento de datos se decidió utilizar ***MySQL*** el cual es un sistema de gestión de base de datos relacional de código abierto más utilizada y documentada para entornos de desarrollo web (cloud-native applications). Esta estará alojada en un droplet en la nube de Digital Ocean y fue considerada por la facilidad de uso y que la cantidad de datos entrantes y salientes no será mucho y, en caso de que esto aumente, se puede escalar de manera fácil ya sea aumentando las capacidades del droplet o replicando la base de datos para aumentar la capacidad de almacenamiento.

- Listado de Microservicios

Carro
Editorial:
Género:
Libro
Usuario
Solicitud
Bitácora

- Definición de servicios web de cada microservicio

USUARIO (fase 1):

- **Signup** (*post*): Permite registrar un nuevo usuario cliente dentro del sistema.

Atributo	Obligatorio	Descripción
nombre	SI	Nombres del usuario
apellidos	SI	Apellidos del usuario
correo	SI	Correo único con el que se registrará
password	SI	Contraseña de la cuenta
teléfono	NO	Numero de telefono de contacto

estado	NO	Estado de actividad o inactividad de la cuenta.

Satisfactorio

```
{
    codigo: 201,
    mensaje: “Insertado correctamente”
}
```

Error

```
{
    codigo: 406,
    mensaje: error.StackTrace
}
```

- **Login Cliente** (*post*): Esta función hace posible el acceso de un cliente al sistema verificando la existencia de sus datos de usuario.

Atributo	Obligatorio	Descripción
usuario	SI	Correo electronico del cliente.
contra	SI	Contraseña de acceso del cliente.

Satisfactorio

```
{
    success: true,
    datos: {
        idc: 2,
        nombres: “Axel Smaylie”,
        apellidos: “López Xum”,
        correo: “axel@gmail.com”,
        telefono: “22446688”,
        estado: 1,
        tipo: 1
    }
}
```

No se encuentra usuario con esas credenciales.

```
{
    success: false
}
```

- **Login Editorial** (*post*): Esta función hace posible el acceso de una editorial al sistema verificando la existencia de sus datos de inicio.

Atributo	Obligatorio	Descripción
usuario	SI	Correo electronico del cliente.
contra	SI	Contraseña de acceso del cliente.

Satisfactorio

```
{
    success: true,
    datos: {
        ide: 2,
        nombre: “Artemis Edinter”
        correo: “tiendaOnlie@artemis.com”,
        direccion: “21 Calle, 14-55 zona 10”,
        estado: 1
    }
}
```

No se encuentra editorial con esas credenciales.

```
{
    success: false
}
```

- **Obtener** (*get*): Obtiene un listado de los datos de los clientes activos.

Atributo	Obligatorio	Descripción
SIN PARÁMETROS DE ENTRADA		

Satisfactorio

```
{
    [
        {
            idc: 1,
            nombres: “....”,
            ...
        },

        ...
    ]
```

}

Error

{

Mensaje: err.StackTrace

}

- **Eliminar** (*post*): Actualiza el estado de un cliente específico de activo a inactivo.

Atributo	Obligatorio	Descripción
id	SI	Id del usuario que se desea eliminar

Satisfactorio

{

success: true,

mensaje: “Eliminado correctamente”

}

No se encuentra editorial con esas credenciales.

{

success: false,

mensaje: err.StackTrace

}

CARRO - EDITORIAL (fase 1):

- ☐ Carrito
- **Insertar** *(post)***:** Esta función añade libros al carrito de compras del cliente.

Atributo	Obligatorio	Descripción
cliente	SI	Id del cliente que compra
libro	SI	Id libro a comprar

Satisfactorio

{

Mensaje: “ok”

}

Error con base de datos.

{
status: 500,
Mensaje: “Error en la consulta”
}

Error con datos obligatorios.

{
status: 422,
Mensaje: “Faltan campos obligatorios”
}

- **Actualizar** *(put)***:** Esta función se utiliza cuando un libro se quite del carrito de compras del cliente.

Atributo	Obligatorio	Descripción
etapa	SI	Valor de la etapa a cambiar.
idr	SI	Id carro a actualizar

Satisfactorio

{
Mensaje: “ok”
}

Error con base de datos.

{
status: 500,
Mensaje: “Error en la consulta”
}

Error con datos obligatorios.

{
status: 422,
Mensaje: “Faltan campos obligatorios”
}

- **Leer** *(get)***:** Devuelve los libros del carrito de compras del cliente.

Atributo	Obligatorio	Descripción
id	SI	Id del carro de compras a leer.

Satisfactorio

{
id_carrito: 2,
id_cliente: 11,
etapa: 1,
...
}

Error con base de datos.

{
status: 500,
Mensaje: “Error en la consulta”
}

Error con datos obligatorios.

{
status: 422,
Mensaje: “Faltan campos obligatorios”
}

☐ Editorial

- **Insertar** *(post)***:** Ingresa una nueva solicitud de editorial para que el administrador la acepte o no.

Atributo	Obligatorio	Descripción
nombre	SI	Nombre de la editorial
correo	SI	Correo electrónico
password	SI	Contraseña de inicio
direccion	SI	Dirección física
estado	NO	Estado de la editorial

Satisfactorio

{
Mensaje: “ok”
}

Error con base de datos.

{
status: 500,
Mensaje: “Error en la consulta”
}

Error con datos obligatorios.

{
status: 422,
Mensaje: “Faltan campos obligatorios”
}

- **Actualizar** *(put)***:** Acepta la solicitud de editorial y estará lista para usarse.

Atributo	Obligatorio	Descripción
id	SI	Id de la editorial a modificar

Satisfactorio
{
Mensaje: “ok”
}

Error con base de datos.
{
status: 500,
Mensaje: “Error en la consulta”
}
Error con datos obligatorios.
{
status: 422,
Mensaje: “Faltan campos obligatorios”
}

- **Eliminar** *(delete)***:** Elimina la editorial seleccionada.

Atributo	Obligatorio	Descripción
estado	SI	Estado a asignar.
ide	SI	Identificador de la editorial

Satisfactorio
{
Mensaje: “ok”
}

Error con base de datos.
{
status: 500,

Mensaje: “Error en la consulta”
}

Error con datos obligatorios.

{
status: 422,
Mensaje: “Faltan campos obligatorios”
}

- **Leer** *(get)***:** Devuelve todas las editoriales para que el administrador pueda gestionarlas.

Atributo	Obligatorio	Descripción
SIN PARÁMETROS DE ENTRADA		

Satisfactorio

{
[
{
ide: 2,
nombre: “Artemis Edinter”,
correo: “oficinaVirtual@artemis.com”,
direccion: “21 Calle, 14-55 zona 10”
...
},

...
]
}

Error

{
Mensaje: “Error en la consulta”
}

GÉNERO (fase 1):

- **Obtener Géneros** (*get*): Servicio que devuelve un listado con todos los géneros literarios almacenados.

Atributo	Obligatorio	Descripción
SIN PARÁMETROS DE ENTRADA		

Satisfactorio

```
{
    [
        {
            idg: 1,
            nombre: “Novela”,
            descripcion: “”
            ...
        },

        ...
    ]
}
```

Error

```
{
    Error: err.message
}
```

LIBRO (fase 1):

- **Crear nuevo libro** (*post*): Crea un nuevo libro dentro de la base de datos asociado a una editorial.

Atributo	Obligatorio	Descripción
nombre	SI	Nombres del libro
autor	SI	Nombre del autor
precio	SI	Precio de venta
cantidad	NO	Cantidad de unidades disponibles
imagen	SI	Imagen de portada
editorial	SI	Referencia a la editorial que crea el libro.
estado	NO	Estado de actividad o inactividad de la cuenta.

Satisfactorio

```
{
    status: 200,
    Mensaje: “ok”
}
```

Error

```
{
    Mensaje: “error”
}
```

- **Obtener todos los libros** (*get*): Obtiene todos los libros activos de todas las editoriales.

Atributo	Obligatorio	Descripción
SIN PARÁMETROS DE ENTRADA		

Satisfactorio

```
{
    [
        {
            idl: 2,
            nombre: “La Odisea”,
            autor: “Homero”
            ...
        },

        ...
    ]
}
```

Error

```
{
    err.message
}
```

- **Eliminar un libro** (delete): Cambia el estado de un libro a inactivo.

Atributo	Obligatorio	Descripción

id	SI	Id del libro que se desea eliminar.

Satisfactorio

{
“El libro ha sido Eliminado”
}

Error

{
err.StackTrace
}

SOLICITUD:

- **Crear** *(post)***:** Con esta función realizamos la creación de una solicitud de libro.

Atributo	Obligatorio	Descripción
nombre	SI	Nombres del libro
autor	SI	Autor del libro
fecha	SI	Fecha de primera publicación
pdf	NO	Libro en formato pdf

Satisfactorio

{
codigo: 200,
mensaje: “ok”
}

Error

{
codigo: 500,
mensaje: error.tryCatch
}

- **Aceptar** *(put)***:** Esta función modifica el estado de una solicitud por aprobada.

Atributo	Obligatorio	Descripción
ids	SI	Id de la solicitud

Satisfactorio

{
codigo: 200,
mensaje: “ok”
}

Error

{
codigo: 500,
mensaje: error.tryCatch
}

- **Leer** *(get)***:** Devuelve todas las solicitudes pendientes de aprobar para que las editoriales las puedan aceptar.

Atributo	Obligatorio	Descripción

Satisfactorio

{
codigo: 200,
results []
}

Error

{
codigo: 500,
mensaje: error.tryCatch
}

BITÁCORA:

- **Insertar** *(post)***:** Función que crea un nuevo registro de una acción hecho por la editorial, como puede ser crear libro, modificarlo, eliminarlo, etc.

Atributo	Obligatorio	Descripción
editorial	SI	id de la editorial
accion	SI	Acción realizada; puede ser crear, modificar, eliminar, etc.
fecha	SI	Fecha en la que se realizó la acción

Satisfactorio

{
codigo: 200,
mensaje: “ok”
}

Error

{
codigo: 500,
mensaje: error.tryCatch
}

- **leerXadmin** *(get)***:** Devuelve todos los logs hechos por las editoriales para que el usuario admin las vea.

Atributo	Obligatorio	Descripción

Satisfactorio

{
codigo: 200,
results []
}

Error

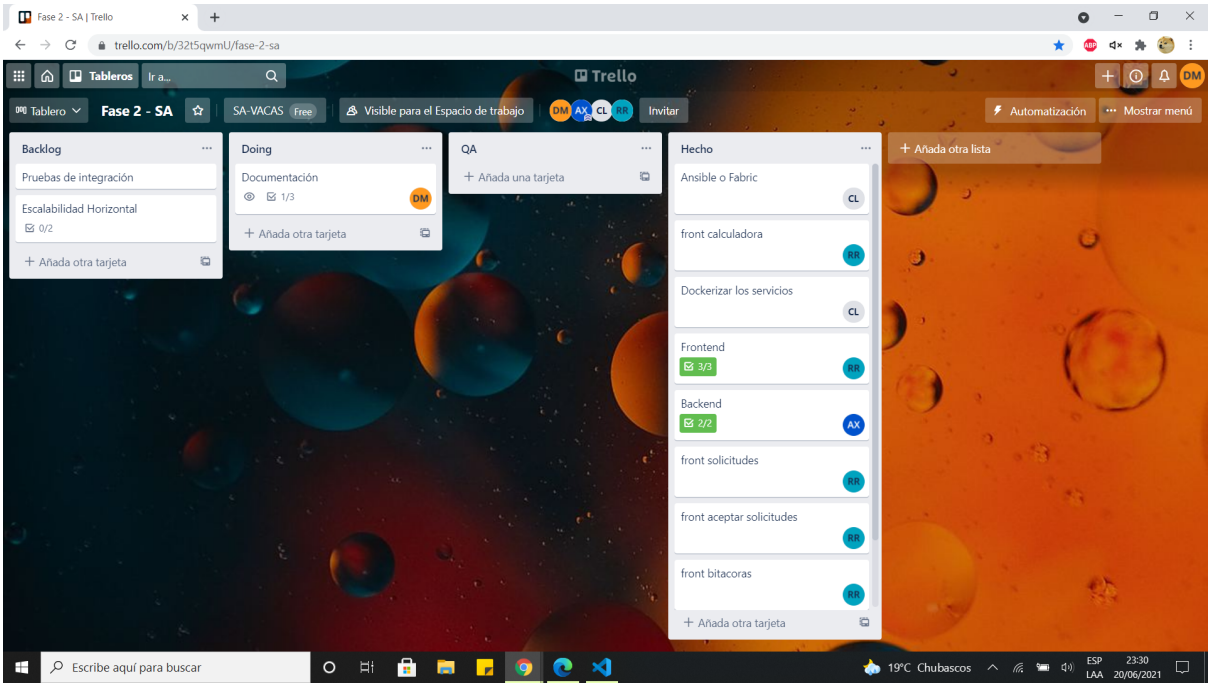
{
codigo: 500,
mensaje: error.tryCatch
}

- Herramienta de metodología a utilizar

***Grupos focales*:** Es una técnica también conocida como focus group donde un grupo reducido de personas interactúan, discuten y elaboran acuerdos mediante la dirección de un moderador; para ello se utilizaron como herramientas de apoyo y de comunicación *Google Meet* y *Whatsapp*.

- Herramientas de control de tiempo de trabajo

En cuanto a la administración del proyecto y también del tiempo de trabajo se utiliza el software de ***Trello*** el cual utiliza los tableros de Kanban para indicar el estado del desarrollo del proyecto de forma visual lo cual facilita la organización y coordinación de los integrantes del equipo de desarrollo. Otro de los beneficios que aporta Trello es que posee una interfaz web y sendas aplicaciones móviles para sistemas iOS y Android para ser más accesible y fácil de usar para los involucrados en el proyecto.



- Pruebas a implementar

Pruebas Unitarias: La función de las pruebas unitarias es comprobar el correcto versionamiento de una unidad de código, sirven para detectar errores en unidades de código de forma temprana ahorrando tiempo y dinero en la resolución de dichos errores en caso de que se detectan en fases más avanzadas del ciclo de vida del proyecto.

Las herramientas para la realización de las pruebas seleccionadas son ***MochaJs*** (framework de pruebas de Node.js), ***Chai JS*** (librería de aserciones para pruebas de JavaScript) para estilizar e identificar mejor el código de las pruebas y por último se utiliza el módulo de Node.js para la prueba de servicios http llamado ***Supertest*** el cual cuenta con la interfaz de Fluent API la cual provee un alto nivel de abstracción tanto en la ejecución de consultas http como en la evaluación de pruebas con los resultados obtenidos de dichas consultas.

services > Genero > test.js > describe('Generos') callback

```
const expect = require('chai').expect;
const request = require('supertest')('http://localhost:3600/api');
process.env.NODE_ENV = 'test';
const app = require('./index');

describe('Generos', () => {
    it('Debe existir editoriales', async function () {
        const response = await request.get('/genero');

        expect(response.body).to.be.an('array');
    });
    it('Debe devolver solo un objeto', async function () {
        const response = await request.get('/genero/1');

        expect(response.body).not.to.be.an('array');
    });
});
describe('Detalle Libro-Genero', () => {
    it('Arreglo de detalles', async function () {
        const response = await request.get('/clasificacion');
        const cuerpo = response.body;

        if(cuerpo.lenght > 0) {
            expect(cuerpo[0]).to.be.an('array');
        } else {
            expect(cuerpo).to.be.an('array');
        }
    });
});
```

Pruebas unitarias servicio de género. ***Total 3***

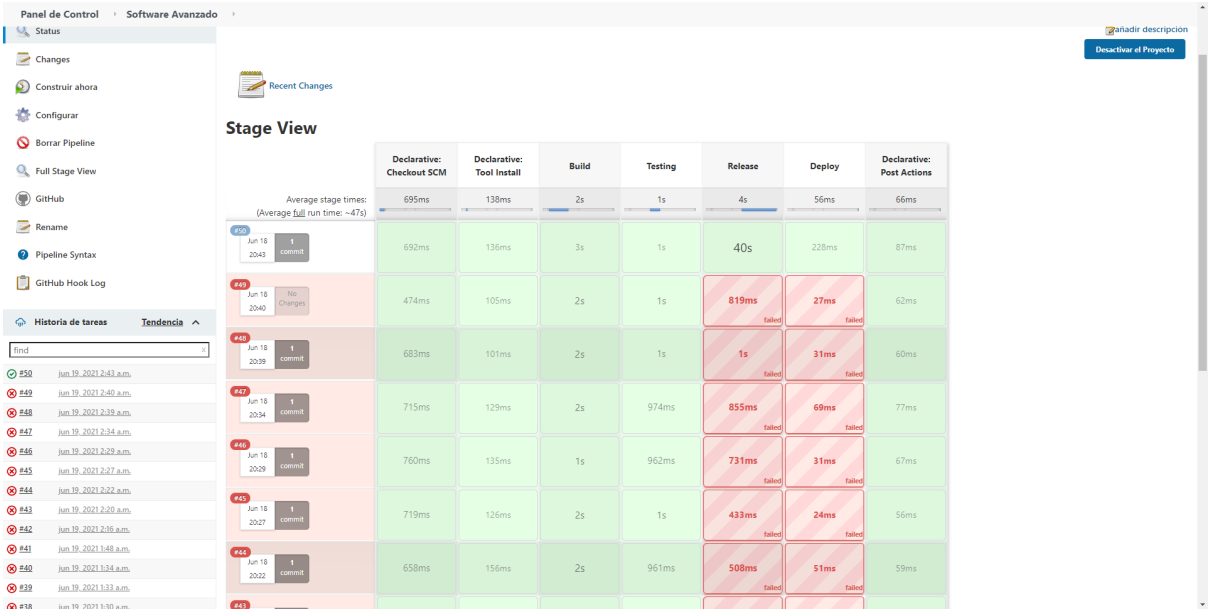
```
const expect = require('chai').expect
const request = require('supertest')('http://localhost:4010/api/editorial')
process.env.NODE_ENV = 'test'
const app = require('./index')

describe('GET /leer', function () {
    it('Debe retornar status 200', async function () {
        const response = await request.get('/leer')

        expect(response.status).to.be.eql(200);
    })
    it('Debe retornar una array', async function () {
        const response = await request.get('/leer')

        const attributes = response.body;
        expect(attributes).to.be.an('array');
    })
    it('La primera editorial debe contener los atributos ide, nombre, correo', async function () {
        const response = await request.get('/leer')

        const attributes = response.body[0];
        expect(attributes).to.include.keys('ide', 'nombre', 'correo');
    })
    it('El campo direccion de la primera editorial debe ser un string', async function () {
        const response = await request.get('/leer')

        const attributes = response.body[0];
        expect(attributes.direccion).to.be.an('string');
    })
    it('tipo de request incorrecto, debe retornar status 404', async function () {
        const response = await request.post('/leer')

        expect(response.status).to.be.eql(404);
    })
})
```

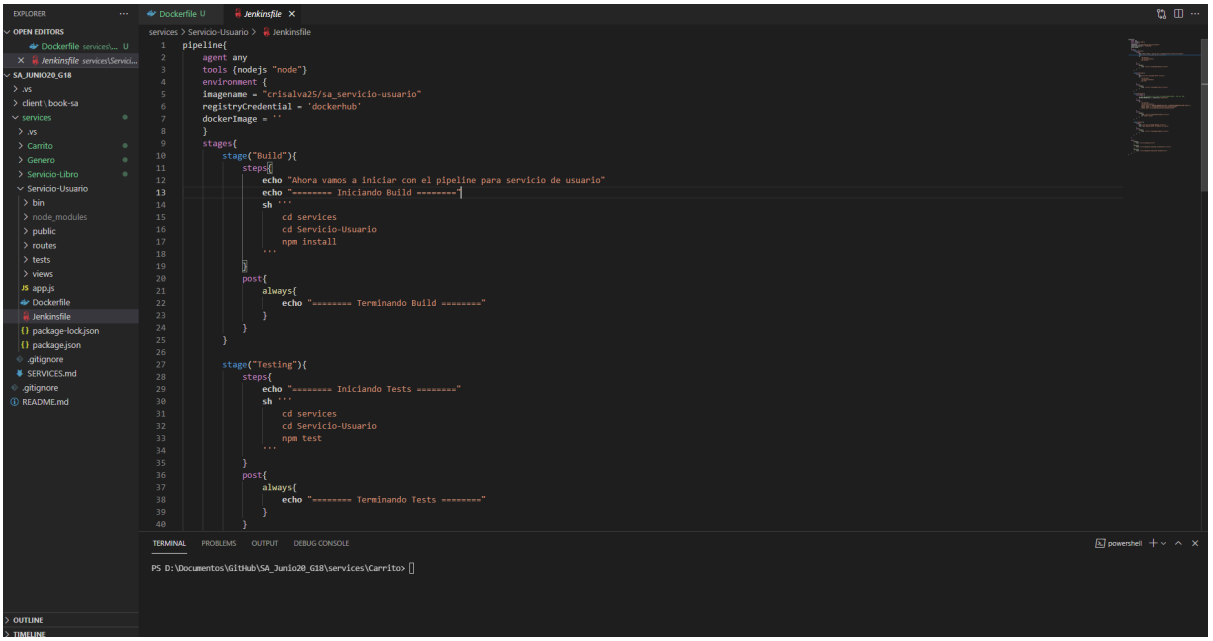
Pruebas unitarias servicio editoriales y carro de compras. ***Total 5.***

Pruebas de Integración: Estas pruebas se realizan posteriormente a las pruebas unitarias y tienen el deber de probar que los elementos unitarios verificado en las pruebas anteriores funcionen correctamente de manera conjunta. Se centra principalmente en probar la comunicación entre componentes (ya sea de hardware o de software).

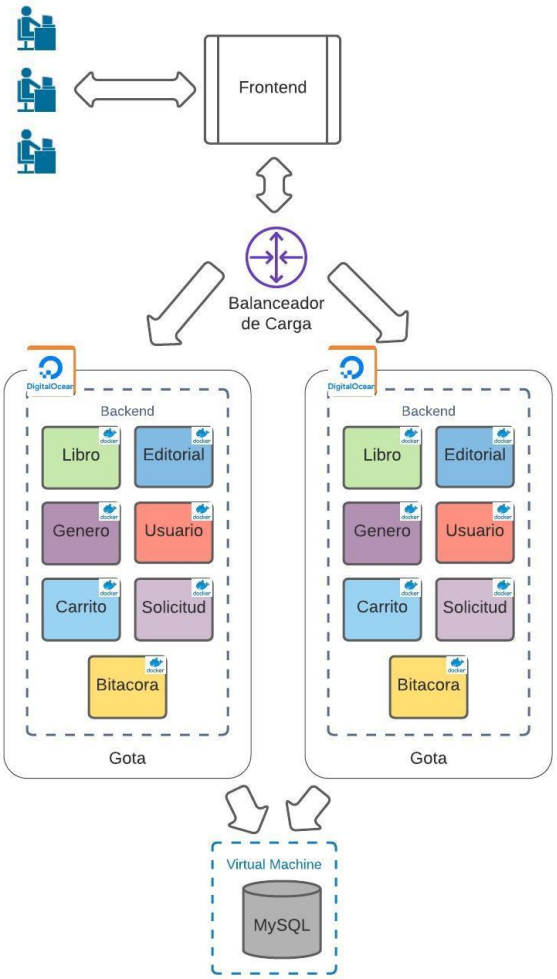
- Herramientas de Configuración - Jenkins

Como herramienta de Configuración utilizamos ***Jenkins***, es un servidor de automatización que ayuda en el proceso de desarrollo de software mediante integración continua y facilita ciertos aspectos de la entrega continua. Admite herramientas de control de versiones como CVS, Subversion, Git, Mercurial, Perforce y Clearcase y puede ejecutar proyectos basados en Apache Ant y Apache Maven, así como secuencias de comandos de consola y programas por lotes de Windows.





- **Arquitectura a implementar**



Arquitectura de Microservicios