Documento de Inspección

Testing de Software

Grupo 5

Walter Renzo Muñoz Arellano 15200089

Cesar Gutierrez Tineo 15200210

## 

## 

## 

## Índice

1. Determinación
2. Productos Software a Revisar
3. Roles
4. Objetivos de la Revisión
   1. Ausencia de Anomalías en los requisitos
   2. Ausencia de Anomalías en el diseño
   3. Ausencia de Anomalías en el código
5. Resultado de la revisión
   1. Proyecto Inspeccionado
   2. Miembros del Equipo / Roles
   3. Duración de la Inspección
   4. Productos Software Inspeccionados
   5. Tamaño del Material Inspeccionado
   6. Anomalías detectadas
      1. a) Incompletitud / Ausencia
      2. b) Extra (superﬂuo)
      3. c) Ambiguo
      4. d) Inconsistente
      5. e) Mejora Deseable
      6. f) No conforme a estándar
      7. g) Propenso a Riesgo
      8. h) Incorrecto
      9. i) No Implementable
      10. j) Editorial

**1.- Determinación**

Encontrar bugs, fallas y errores que tenga el software desarrollado por el grupo 5 para así darle un grado de aprobación y saber si está listo para el pase a producción, esto se llevará a cabo con la técnica de inspección de código y también con la ayuda del software open source SonarQue.

**2.- Productos de Software a Revisar**

**Frontend**

Interfaces de usuario programadas para cubrir la gestión de una biblioteca tanto para estudiantes como para encargados.

Funcionalidades observadas:

1. Búsqueda de libros.
2. Separación de un libro.
3. Aceptación de solicitud.
4. Historial de acciones
5. Registro de Editorial
6. Registro de Autores
7. Ingreso de un nuevo libro.
8. Eliminación de un libro.
9. Estadísticas.
10. Registro de Alumnos.
11. Login. (Google validation opcional).
12. Landing de Inicio.

**Backend**

Software desarrollado para recibir peticiones y se encuentra directamente conectado a la Base de datos del proyecto.

Funcionalidades observadas:

Conexión a DB (Firebase)

Solicitudes POST,GET,PUT,DELETE

**Tecnologías utilizadas:**

1. AngularJS
2. Firebase (Cloud DB)
3. NestJS
4. HTML5
5. CSS
6. Javascript
7. MongoDB
8. NPM

**3.- Roles**

Testers a cargo:

1.- Walter Renzo Muñoz Arellano

Tareas:

* **Lista de chequeo para el documento de especificación de requisitos**
* **Matriz de Inspección**

1.- Cesar Gutierrez Tineo

Tareas:

* **Lista de chequeo de estándares de funciones y procedimientos**
* **Checklist del código especializado en el lenguaje**

1.- Juan Alonso Arce

Tareas:

* **Lista de chequeo de estándares de programación - código**
* **Matriz de Manual de Usuario**

1.- Bryan Lázaro Cusihuaman

Tareas:

* **Checklist: Manual de Instalación**
* **Lista de chequeo de inspección de código para el lenguaje de programación**

**4.- Objetivos de la Revisión**

Los objetivos planteado corresponden a que se espera que el software entregado para estas pruebas tiene que pasar cada una de estas para así pasar a una etapa de producción.

Cada una de estas anomalías planteadas corresponde.

**4.1 Ausencia de Anomalías en los requisitos**

Este punto aún es incierto ya que no se llegó a proporcionar un documento de requisitos ni tampoco historias de usuario por lo que se tuvo que realizar pruebas de manera “práctica” es decir, solo con la experiencia del tester encargado con respecto al funcionamiento de una biblioteca.

**4.2 Ausencia de Anomalías en el diseño**

Se tomará como diseño a las interfaces desarrolladas en el software (Frontend) y se tiene en cuenta la sencillez de uso, colores y tiempo de espera con respecto a carga de datos y demás para así llegar a una conclusión.

**4.3 Ausencia de Anomalías en el código**

Con las pruebas de testing para el código se quiere llegar a que el producto de software a revisar cumpla los estándares según el lenguaje de programación utilizado así como que se haya hecho uso de patrones de software.

**5.- Resultados de la solución**

1. **Proyecto Inspeccionado**

“LibFISI” - Proyecto desarrollado como sistema de biblioteca para la Facultad de Sistemas e Informática

1. **Miembros del Equipo / Roles**

* ANTAURCO ARAGÓN, Gabriel Giancarlo 16200204
* BARBIERI LIZAMA, Giordano 16200251
* CASAS RIME, Jordan Eddy Brandon 16200058
* PEREZ LOPEZ, Juan Carlos 12200197

1. **Duración de la Inspección**

Tiempo previsto : 7 días hábiles

1. Productos Software Inspeccionados

* Código Fuente del Backend
* Código Fuente del Frontend
* Documento de Modelo de Inventario
* Manual de Instalación
* Manual de Usuario

1. Tamaño del Material Inspeccionado

**Datos obtenidos del Modelo de Inventario - “LibFISI”**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | **Código Fuente del Backend** | Repositorio de Github | 1 carpeta(5 carpetas, 13 archivos) | Este carpeta contiene:  github/ISSUE\_TEMPLATE: Este archivo se utiliza como plantilla de problemas.   * bug\_report.md: Una descripción clara y concisa de lo que es el error.   archivos: Este archivo se utiliza para guardar las IPs de los clientes.   * ipCliente.txt: Contenedor de IPs   fotos: Este archivo contiene fotos de prueba.   * foto1.png * foto3.png   src /archivos: Esta carpeta contiene controladores con sus respectivos servicios.   * archivos.controller.spec.ts: Contiene la finalización Modelo Primitivo Usuarios y API Archivos. * archivos.controller.ts: Contiene el mensaje de error en middleware. * archivos.module.ts: Contiene el código para la sobre escritura del @ Módulo de @ nestjs/common con MongooseModule de @ nestjs/mongoose, ArchivosController de archivos.controller, ArchivosService de archivos.service y ArchivoSchema de esquemas/archivo.schema. * archivos.service.spec.ts: Contiene la finalización Modelo Primitivo Usuarios y API Archivos. * archivos.service.ts: Contiene mensajes de error formato json. * ipCliente.txt: Contiene el Análisis de ipUsuarios.   src /autores: Esta carpeta contiene archivos que permitirán gestionar los autores de los libros.   * autores.controller.ts; Contiene código que permite gestionar los libros. * autores.module.ts: Contiene código que permite gestionar los autores de los libros. * autores.service.ts: Contiene código para la gestión de editoriales.   src /componentes: Esta carpeta contiene las carpetas castigos, pedido, préstamos y temas.   * Castigos: Esta carpeta contiene archivos que ayudan a gestionar los castigos por incumplimiento de las reglas establecidas.  1. castigos.controller.ts: Contiene el código del controlador de castigos. 2. castigos.module.ts: Contiene el código para el servicio castigar usuario 3. castigos.service.ts: Contiene el código que permite poner la sanción a usuario.  * Pedidos: Esta carpeta contiene archivos que ayudan a gestionar los pedidos de usuarios.  1. pedidos.controller.ts: Contiene el código para manejar los pedidos. 2. pedidos.module.ts: Contiene el código para obtener los pedidos de un usuario. 3. pedidos.service.ts: Contiene el código para gestionar el pedido.  * Prestamos: Esta carpeta contiene archivos que ayudan a gestionar los préstamos de los usuarios.  1. prestamos.controller.ts: Contiene el código para analizar la fecha de fin de préstamo. 2. prestamos.module.ts: Contiene el código para los servicios que permiten realizar los préstamos por un usuario. 3. prestamos.service.ts: Contiene el código para gestionar el préstamo.  * Tema: Esta carpeta contiene los archivos que permiten manejar los temas.  1. temas.controller.ts: Contiene el código para manejar los temas de los libros. 2. temas.module.ts: Contiene el código para relacionar los libros con Temas. 3. temas.service.ts: Contiene el código que permiten arreglar los temas.   src /editoriales: Esta carpeta contiene los archivos que permiten gestionar las editoriales.   * editoriales.controller.ts: Contiene el código para manejar las editoriales. * editoriales.module.ts: Contienen el código para gestionar las editoriales. * editoriales.service.ts: Contiene le código que permite modificar o eliminar editoriales.   src /esquemas: Esta carpeta contiene los archivos de esquemas.   * archivo.schema.ts: Contiene el código que permite obtener el esquema de mongosta. * autor.schema.ts: Contiene el código que determina qué datos se necesita de los autores. * castigo.schema.ts: Contiene el código que permite aplicar los castigos al usuario. * cuenta.schema.ts: Contiene el código que permite capturar los datos de usuarios con cuentas de Google. * editorial.schema.ts: Contiene el código que permite gestionar editoriales. * item.schema.ts: Contiene el código que permiten gestionar los Artículos y la relación de Autores y Editoriales. * libro.schema.ts: Contiene el código que determinar qué datos se necesita de los libros. * libro\_autor.schema.ts: Contiene el código que permite relacionar libro y autor. * libro\_editorial.schema.ts: Contiene el código que permite relacionar libro y editorial. * libro\_tema.schema.ts: Contiene el código que permite relacionar el libro con los temas. * pedido.schema.ts: Contiene el código que permite asignar los datos a los pedidos. * tema.schema.ts: Contiene el código que permite relacionar los temas. * tipoUsuario.schema.ts: Contiene el código que permite especificar el tipo de usuario. * usuario.schema.ts: Contiene el código que determinar qué datos se necesita del usuario.   src /interfaces: Este archivo contiene las interfaces .   * Archivo.ts: * Autor.ts * Castigo.ts * Cuenta.ts * Editorial.ts * Item.ts * Libro.ts * Libro\_Autor.ts * Libro\_Editorial.ts * Libro\_Tema.ts * Pedido.ts * Prestamo.ts * Tema.ts * TipoUsuario.ts * Usuario.ts   src /libros: Esta carpeta contiene archivos que permiten gestionar los libros.   * libros.controller.ts: Este archivo contiene el código para controlar los libros. * ibros.module.ts: Este archivo contiene el código para modular los libros. * libros.service.ts: Este archivo contiene el código para el paginado de libros frontend.   src /middlewares: Esta carpeta contiene el archivo que permite autenticar una cuenta.   * logeador.middleware.ts: Este archivo contiene el código para autenticar una cuenta.   src /modelos: Esta carpeta contiene un archivo con Api Usuarios con modelo de prueba.   * usuario.dto.ts: Este archivo contiene un Api de Usuarios con modelo de prueba.   src /tipoUsuarios: Esta carpeta contiene archivos que permiten gestionar los tipos de usuarios.   * tipoUsuarios.controller.ts: Este archivo nos permite controlar los tipos de usuarios. * tipoUsuarios.module.ts: Este archivo nos permite modular los tipos de usuarios. * tipoUsuarios.service.ts: Este archivo nos permite modificar o eliminar los tipos de usuarios.   src /usuarios: Esta carpeta contiene los archivos que nos permiten gestionar los usuarios con google.   * usuarios.controller.ts: Este archivo nos permite controlar los usuarios con google. * usuarios.module.ts: Este archivo nos permite modular los usuarios con google. * usuarios.service.ts: Este archivo nos permite modificar o eliminar los usuarios con goolgle.   Src:   * app.controller.spec.ts * app.controller.ts * app.module.ts * app.service.ts * main.ts   prueba: Esta carpeta contiene los archivos de pruebas   * app.e2e-spec.ts * jest-e2e.json   .gitignore: Este archivo permite gestionar autores.  .prettierrc  README.md  SEGURIDAD.md  nest-cli.json  nodemon-debug.json  nodemon.json  package-lock.json  package.json  prueba.js  tsconfig.build.json  tsconfig.json  tslint.json |
| **2** | **Código Fuente del Frontend** | Repositorio de Github | 1 carpeta(4 carpetas, 9 archivos) | Esta carpeta contiene:  Dist: Esta carpeta contiene el archivo prueba.   * prueba.html: Este archivo contiene el código para manejar la búsqueda de Libros para los usuarios.   e2e: Esta carpeta solo contiene vistas.   * protractor.conf.js * tsconfig.e2e.json   login: Esta carpeta contiene los archivos para autenticar el login.   * login.component.css * login.component.html * login.component.spec.ts * login.component.ts   src: Esta carpeta contiene archivos y carpetas para cambiar nombres.   * App: cambio de nombre * Assets: mejora del frontend * Environments * Examples: esta carpeta solo contiene vistas. * Browserslist * favicon.ico * index.html: mejora el login y la paginación * karma.conf.js * main.ts * polyfills.ts * styles.css * test.ts * tsconfig.app.json * tsconfig.spec.json * tslint.json   .editorconfig  .gitignore  README.md  angular.json  package-lock.json  package.json  server.js  tsconfig.json  tslint.json |
| **3** | **Manual de Instalación y Configuración** | Documento Word | 1 Documento | Este documento contiene:  Presentación  Alcance de este manual  Requerimientos técnicos   * Requerimientos mínimos de hardware * Requerimientos mínimos de hardware   Herramientas utilizadas para el desarrollo  Instalación   * Base de datos MongoDB Atlas * Node.JS * Nest (NestJS) * Angular JS * Configuración * Base de datos MongoDB * Angular y MongoDB atlas |
| **4** | **Manual de Usuario** | Documento Word | 1 Documento | Este documento contiene:  Introducción  Visión general de la aplicación web Trick Reading  Pantalla de Inicio  Log in  Opciones como Alumno  Búsqueda de Libros  Buscar un libro especifico  Solicitar Libro  Confirmación de solicitud  Estado del préstamo solicitado  Concurrencia de préstamos  Historial de préstamos  Perfil del Alumno  Salida del sistema  Opciones como Administrador  Pantalla de Inicio  Gestión de libros  Registrar Alumno  Visualizar listado de alumnos registrados  Ver información del alumno  Editar información del alumno  Eliminar alumno  Gestión de libros  Gestión de editoriales  Creación de nuevas editoriales  Eliminar editorial  Gestión de autores  Editar autor  Eliminar autor  Gestión de prestamos  Aceptar solicitud  Rechazar solicitud  Gestión de estadísticas |

1. **Anomalías detectadas**

Todas las anomalías detectadas han sido especificadas en el documento de tipo Excel llamado **Matriz de Inspección**