|  |
| --- |
| **Arquitectura Tecnológica del Sistema** |
| **Proyecto: APLICATIVO MÓVIL PARA EL AUTOPRÉSTAMO DE LIBROS EN LA BIBLIOTECA BENEDICTO XVI**  Revisión 1 |

**FICHA DEL DOCUMENTO**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Revisión** | **Autor** | **Revisado Por** |
|  | 01 | Alexandra Bernal Escalante  Nicolle Fabiana Cadavid Noguera | Elkin Alfredo Albarracín Navas |

1. ESTRUCTURA DEL SISTEMA

|  |
| --- |
| DIAGRAMA DE DESPLIEGUE |
|  |
| ESPECIFICACIÓN |
| |  |  | | --- | --- | | DESCRIPCIÓN | El diagrama de despliegue representa los diferentes componentes que conforman al sistema y en los nodos en los que se encuentran, como también sus interfaces y protocolos de comunicación. | | RESPONSABILIDADES PRINCIPALES | El sistema de autopréstamo está conformado por dos clientes, uno siendo el **aplicativo móvil** desarrollado por medio del framework *flutter* en su versión 3.32.8 y la **aplicación de escritorio** desarrollada con *java* en su versión 22 y *javafx en su versión 24.0.2*. Cada uno los clientes se comunican con el **backend** por medio de una API REST a través del protocolo HTTPS. El componente del **Backend** se ejecuta a través del entorno de ejecución *NodeJS* en su versión 22.16.0, este se encuentra desplegado en una máquina virtual (VM) con un sistema operativo Red Hat Enterprise Linux en su versión 9.6.  A través del protocolo TCP/IP el **backend** se comunica con un *Database Management System* (DBMS) MySQL, en su versión 5.0.51a, para poder acceder a la base de datos “biblioteca” | | OBSERVACIÓN | La infraestructura del sistema se encuentra implementada bajo la gestión y lineamientos del departamento CTIC de la Universidad Pontifica Bolivariana seccional Bucaramanga. Esta puede estar sujeta a modificaciones de acuerdo con la planeación y necesidades de dicho departamento. | |

1. COMPORTAMIENTO DEL SISTEMA

|  |  |
| --- | --- |
| DIAGRAMAS DE COMPORTAMIENTO – MÓDULO AUTOPRÉSTAMO | |
| DIAGRAMA DE SECUENCIA – SOLICITAR PRESTAMO | |
|  | |
| DESCRIPCIÓN | El siguiente diagrama de secuencia representa el proceso general que se realiza en el backend cuando desde la aplicación móvil, se hace la solicitud de un préstamo. |
| ESPECIFICACIONES | |
| |  |  | | --- | --- | | RESPONSABILIDAD | | |  | El aplicativo móvil envía una petición HTTP a través de un endpoint identificado con la ruta /**solicitudPrestamo,** definido en el **Router.**  Para poder acceder a cada uno de los API endpoint, es necesario pasar por el **AuthMiddleware*,*** encargado de validar la vigencia del token JWT enviado en la petición HTTP, por medio del header *Authentication* con la estrategia Bearer, en el caso que exista algún tipo de inconveniente, se retornan un estado HTTP, junto a un mensaje, en donde se indica que el usuario no puede ser autenticado para usar la ruta. Si el token es válido, el flujo de la petición continua, en este caso se invoca el método *validarSolicitudPrestamo del* **AutoprestamoController.** | |  | Denominado por **consultarValidaciones,** hace referencia a cada uno de los métodos que se invocan en **AutoprestamoService** pertenecientes al repositorio, estos siendo *consultarLibro, tipoColeccion, consultarLibroPrestado, consultarMultas, calcularPrestamosActivos,*  **AutoConsultasRepository,** encargado de obtener la información de la base de datos **biblioteca** a través de la clase *MySqlDBC,* esta retorna una variable genérica T con la data, la cual se utiliza para verificar por medio de condicionales que el préstamo se puede realizar. | |  | Si el préstamo se puede realizar el siguiente paso es buscar la información del libro en la base de datos, para posteriormente retornarla al controlador por medio de una respuesta predeterminada por la interfaz **PrestamoResponse** y por último que este retorne una respuesta en formato JSON, junto con un código de estado HTTP | | OBSERVACIONES | | | El proceso de validar se especifica a más detalle en el diagrama de flujo *validarPrestamo*. Cada una de las clases, métodos y variables se especifican en el diagrama de clases descrito con anterioridad. | | | |
| DIAGRAMA DE SECUENCIA- REALIZAR PRESTAMO | |
|  | |
| DESCRIPCIÓN | El siguiente diagrama de secuencia representa el proceso que se implementa cuando se confirma la realización de un préstamo, es decir calcular el respectivo tiempo y guardar la información en la base de datos. |
| ESPECIFICACIONES | |
| |  |  | | --- | --- | | RESPONSABILIDAD | | |  | Después de que al cliente se le valida la solicitud del préstamo y confirma que quiere realizarlo, el aplicativo móvil por medio de la ruta **/prestamo** definida en el **Router** envía la solicitud para realizarlo. De primer momento como en todas las solicitudes, se verifica que el usuario se encuentra autenticado por medio del token, en el caso que no, el servidor devuelve un *HTTP status 401* y se termina la conexión.  Por otra parte, si el token es válido el flujo de la petición continua, en este caso se invoca el método *realizarPrestamo* de la clase **AutoprestamoController** y así mismo el controller llama a *guardarPrestamo* perteneciente a **AutoprestamoService,** allí se realiza la validación que el libro no se encuentra prestado por medio de **AutoConsultasRepository** encaragdo de realizar la consulta a la base de datos. | |  | En el caso que el libro no se encuentre prestado **AutoprestamoService** utiliza el método *obtenerDíasNohabiles* de la clase **AutoConsultasRepository** para obtener de la base de datos los días que no se tienen en cuenta para calcular el tiempo del préstamo, posteriormente se invoca la función *calcularTiempo* que retorna la fecha de inicio y fin del préstamo.  **AutoGuardarRepository** es el encargado de guardar los datos en las tablas referentes al préstamo usando la clase MySqlDBC a través de consultas usando INSERT. Para el método *guardarLog,* es necesario verificar que el atributo ipPublica viene en el cuerpo de la petición.  Finalmente, el servicio le retorna al controlador un repuesta predeterminada con la interfaz *AppResponsePrestamo* y el controlador retorna una respuesta al aplicativo móvil en formato JSON, junto con un código de estado HTTP. | | OBSERVACIONES | La descripción de cada una de las clases junto a sus métodos y atributos correspondientes, como las interfaces, se pueden observar en el diagrama de estructura. | | |

|  |  |
| --- | --- |
| DIAGRAMA DE COMPORTAMIENTO – MÓDULO DE CONSULTAR PRESTAMOS | |
|  | |
| DESCRIPCIÓN | El siguiente diagrama de secuencia representa el proceso para obtener los prestamos activos del usuario. Esta contextualizado en su implementación en el backend. |
| ESPECIFICACIONES | |
| |  |  | | --- | --- | | RESPONSABILIDAD | El aplicativo móvil por medio del API endpoint **/prestamosActivos/:userID** solicita al servidor la lista de los prestamos activos del usuario, este edpoint como los demás, cuenta con un middleware de autenticación implementado en la clase **AuthMiddleware encargado** de validar el token, en caso de que no sea válido o exista algún problema este retorna un *HTTP status 401 j*unto a un mensaje.  **PrestamoService** se encarga de invocar el método *prestamosActivos* de la clase **PrestamoRepository**, el cual se encarga de llamar a **MySqlDBC** encargada de buscar en la base de datos **biblioteca** los prestamos activos del usuario, si el usuario tiene prestamos activos, a cada uno de los préstamos por medio del método *calcularDiasRestantes,* se les determina los días que faltan para que el préstamo se termine.  El servicio le retorna al controlador una respuesta predeterminada con la interfaz *AppResponseInfoPrestamo,* para que este le retorne al aplicativo móvil una respuesta conformada por un código de estado HTTP y un objeto en formato JSON. | | OBERVACIONES | La descripción de cada una de las clases junto a sus métodos y atributos correspondientes, como las interfaces, se pueden observar en el diagrama de estructura. | | |

|  |  |
| --- | --- |
| DIAGRAMA DE COMPORTAMIENTO – MÓDULO DE AUTENTICACIÓN | |
|  | |
| DESCRIPCIÓN | El siguiente diagrama de secuencia representa el proceso de autenticación del usuario. Esta contextualizado en su implementación en el backend. |
| ESPECIFICACIONES | |
| |  |  | | --- | --- | | RESPONSABILIDAD | El aplicativo móvil por medio del API endpoint **/login** solicita realizar el proceso de inicio de sesión, para esto debe ingresar el usuario y contraseña. Posteriormente se busca la contraseña y si esta coincide con la contraseña ingresada, se genera un JWT por medio del **JWTManager** que se utiliza para autenticarse en peticiones posteriores.  El servicio le retorna una respuesta predeterminada al controlador determinada en la interfaz *AppResponseAuth,* en donde si el atributo *succes* esta en *false* significa que no se pudo iniciar sesión.  Finalmente, el controlador le retorna una respuesta al aplicativo móvil, formada por un código de estado HTTP y un objeto en formato JSON. | | OBERVACIONES | La descripción de cada una de las clases junto a sus métodos y atributos correspondientes, como las interfaces, se pueden observar en el diagrama de estructura. | | |