# ACTIVIDADES DESARROLLADAS DURANTE LA PASANTÍA

1. Implementar la opción de “Menú Avanzado” en el menú principal, a partir de la cual se pueden agregar paquetes y visualizar los ya existentes.

Para agregar paquetes, se crea una ruta que dirige al usuario al método “createPack”, ubicado en el controlador “packageController” que permite mostrar la vista “create.blade” que contiene una ventana modal con el formulario de agregar paquetes, con los siguientes campos:

* Nombre
* Dirección URL del manual
* Idioma
* Categorías
* Descripción

Dentro del formulario se realizan las siguientes validaciones JS (JavaScript):

* El campo “Nombre” no puede estar vacío y sólo puede contener caracteres alfanuméricos y sub-guiones.
* El campo “Dirección URL del manual” no puede estar vacío.
* El campo “Descripción” no puede estar vacío.

Una vez se verifica la validez de la información introducida en el formulario, se procede a utilizar el método “create” del modelo PackageManager para acceder a la tabla “packages” de la base de datos MySQL“ivichem-release” y almacenar en ella un nuevo registro, así como también se procede a crear una base de datos SQLite que se encargará de almacenar información posterior relacionada con el paquete.

Para visualizar los paquetes existentes, se elabora una ruta que permite, a través del método “viewPackages” del controlador “PackageController”, mostrar la vista “view\_packages” que renderiza la información de los paquetes asociados al usuario a través del modelo “Package”

1. En función de su tipo de contribución, el usuario podrá ejecutar determinadas operaciones sobre los paquetes.

* Creador y editor: Pueden agregar colaboradores, editar interface, actualizar y eliminar paquetes.
* Traductor y revisor: Pueden editar interface.

Para realizar cada una de estas operaciones sobre los paquetes, se realizan las siguientes acciones:

* Agregar colaboradores: mediante una ruta, el usuario se dirige al método “addContributorPack” del controlador “PackageController”, el cual muestra una vista que contiene una ventana modal con un formulario con los campos:
  + Nombre del colaborador: es una caja de texto que muestra posibles colaboradores en función de lo que introduce el usuario, para ello se realizó una conexión con la base de datos que permite obtener datos a ser mostrados a través de JS.
  + Rol: es un selector que contiene los roles que puede adquirir un usuario en relación a un paquete (editor, traductor y revisor).

Una vez enviado el formulario, se almacena la información en la tabla MySQL“contributors” y se envía una notificación al usuario a quien se envió la petición de colaboración, quien tiene la opción de aceptarla o rechazarla. En caso de aceptarla, se habilita el registro, y en caso contrario, se elimina.

* Actualizar paquete: mediante una ruta, el usuario se dirige al método “updatePack” del controlador “PackageControl”, el cual se encarga de abrir el formulario de la vista “update\_pack.blade” en una ventana modal. Dicho formulario es llenado automáticamente con los valores guardados en la tabla, a través de una consulta a la tabla “packages” y “categories”. Una vez se hacen las modificaciones, el usuario envía la información, y mediante el método “updatePack” del modelo “PackageManager” se hacen los cambios en la base de datos.
* Eliminar paquete: Mediante una ruta, el usuario se dirige al método “deletePack” del controlador “PackageControl”, el cual se encarga de ejecutar el método “delete” del modelo “PackageManager” que permite eliminar el registro deseado en la base de datos.
* Editar interface: Mediante una ruta, el usuario se dirige a la vista “edit\_interface” a través del método “edit” del controlador “InputEditionController”. Esta vista contiene en forma estructurada las pestañas, secciones, controles y opciones contenidas en cada paquete. Dentro de este entorno se pueden realizar las operaciones CRUD sobre cada elemento.

1. Las pestañas requieren de un archivo asociado, por lo que se implementa un botón que contiene la ruta que dirige al método “newFileType” del controlador “PackageController”, el cual muestra la vista “new\_file\_type” que contiene una ventana modal con el formulario que solicita los datos necesarios para la creación de un nuevo archivo, los cuales son:

* Clasificación: selector de opciones “input” y “output”
* Extensión: caja de texto
* Constante: checkbox, si está marcado, aparecen los campos “prefijo” y “sufijo”, en caso contrario, aparece el campo “Nombre constante”.
* Prefijo: Caja de texto
* Sufijo: Caja de texto
* Nombre constante: Caja de texto

Además de mostrar los campos, contiene una sección que muestra dinámicamente el nombre del paquete en función de lo que el usuario introduce en tiempo real, a modo de referencia.

El formulario tiene las siguientes validaciones JS:

1. Los campos “extensión”, “prefijo” y “sufijo” son requeridos y deben ser alfanuméricos.

Una vez enviado el formulario, se ejecuta el método “addNewFileType” del modelo “PackageManager” el cual llena la tabla “file\_types” de la base de datos SQLite del paquete.

* Agregar pestaña: se crea un botón con una ruta al método “newTab” del controlador “inputEditionController”, el cual hace conexión con la base de datos SQLite del paquete y retorna la vista “new\_tab”, la cual contiene una ventana modal con un formulario con los siguientes campos:
  + Nombre: Caja de texto, la cual tiene una validación JS que lo convierte en campo requerido, alfanumérico, de longitud no menor de 5 caracteres y no mayor de 128 caracteres
  + Palabra clave: Caja de texto, la cual tiene una validación JS que lo convierte en campo requerido, no mayor de 128 caracteres.
  + Archivo asociado: Selector, el cual permite seleccionar a alguno de los archivos de tipo “entrada” creados en el paquete.
  + Control asociado: Caja de texto, la cual muestra en forma dinámica todas las secciones y/o controles posibles mediante la comparación de lo escrito en el campo con la información almacenada en la base de datos.
  + Opción asociada: Caja de texto, está escondida por defecto, y aparece cuando es seleccionado un control asociado. muestra en forma dinámica las opciones que pertenecen al control seleccionado mediante la comparación de lo escrito en el campo con la información almacenada en la base de datos, y permite autocompletar mediante la selección de una opción.

Una vez enviado el formulario, se accede al método “saveTab” del modelo “Package”, el cual almacena un nuevo registro en la tabla “tabs” de la base de datos SQLite asociada al paquete.

* Modificar pestaña: se crea un botón con una ruta al método “modifyTab” del controlador “inputEditionController”, el cual hace conexión con la base de datos SQLite del paquete y retorna la vista “modify\_tab”, la cual contiene una ventana modal con un formulario con los mismos campos y validaciones utilizados en la vista “new\_tab”. Una vez enviado el formulario, se accede al método “modifyTab” del modelo “Package”, el cual modifica el registro en la tabla “tabs” de la base de datos SQLite asociada al paquete, mediante el ID de la pestaña.
* Información: se crea un botón con una ruta al método “infoTab” del controlador “inputEditionController”, el cual hace conexión con la base de datos SQLite del paquete y retorna la vista “info\_tab”, la cual contiene una ventana modal con información de la pestaña, estructurada de la siguiente forma:
  + Nombre
  + Palabra clave
  + Archivo asociado
  + Sección asociada
  + Control asociado
  + Opción asociada
* Eliminar: se crea un botón que acciona una ventana modal de confirmación con una ruta al método “deleteTab” del controlador “inputEditionController”, el cual utiliza el método “deleteTab” del modelo “Package” para eliminar el registro de la base de datos SQLite asociada al paquete.

1. Secciones: se maneja el CRUD de la siguiente forma:

* Agregar: se crea un botón con una ruta al método “newSection” del controlador “inputEditionController”, el cual hace conexión con la base de datos SQLite del paquete y retorna la vista “new\_section”, la cual contiene una ventana modal con un formulario con los siguientes campos:
  + Nombre: Caja de texto, la cual tiene una validación JS que lo convierte en campo requerido, alfanumérico, de longitud no menor de 5 caracteres y no mayor de 128 caracteres
  + Palabra clave: Caja de texto, la cual tiene una validación JS que lo convierte en campo requerido, no mayor de 128 caracteres.
  + Control asociado: Caja de texto, la cual muestra en forma dinámica todas las secciones y/o controles posibles mediante la comparación de lo escrito en el campo con la información almacenada en la base de datos.
  + Opción asociada: Caja de texto, está escondida por defecto, y aparece cuando es seleccionado un control asociado. muestra en forma dinámica las opciones que pertenecen al control seleccionado mediante la comparación de lo escrito en el campo con la información almacenada en la base de datos, y permite autocompletar mediante la selección de una opción.

Una vez enviado el formulario, se accede al método “saveSection” del modelo “Package”, el cual almacena un nuevo registro en la tabla “sections” de la base de datos SQLite asociada al paquete.

* Modificar: se crea un botón con una ruta al método “modifySection” del controlador “inputEditionController”, el cual hace conexión con la base de datos SQLite del paquete y retorna la vista “modify\_section”, la cual contiene una ventana modal con un formulario con los mismos campos y validaciones utilizados en la vista “new\_section”. Una vez enviado el formulario, se accede al método “modifySection” del modelo “Package”, el cual modifica el registro en la tabla “sections” de la base de datos SQLite asociada al paquete, mediante el ID de la pestaña.
* Información: se crea un botón con una ruta al método “infoSection” del controlador “inputEditionController”, el cual hace conexión con la base de datos SQLite del paquete y retorna la vista “info\_section”, la cual contiene una ventana modal con información de la pestaña, estructurada de la siguiente forma
  + Nombre
  + Palabra clave
  + Control asociado
  + Opción asociada
* Eliminar: se crea un botón que acciona una ventana modal de confirmación con una ruta al método “deleteSection” del controlador “inputEditionController”, el cual utiliza el método “deleteSection” del modelo “Package” para eliminar el registro de la base de datos SQLite asociada al paquete.

1. Controles: Se maneja el CRUD de la siguiente manera:

* Agregar: se crea un botón con una ruta al método “newControl” del controlador “inputEditionController”, el cual hace conexión con la base de datos SQLite del paquete y retorna la vista “new\_control”, la cual contiene una ventana modal con un formulario con los siguientes campos:
  + Nombre: Caja de texto, la cual tiene una validación JS que lo convierte en campo requerido, alfanumérico, de longitud no menor de 5 caracteres y no mayor de 128 caracteres
  + Tipo de control: Selector, con las opciones:
    - Checkbox
    - Selector
    - Caja de texto
    - Area de texto
  + Ayuda: Área de texto.
  + Palabra clave: Caja de texto, la cual tiene una validación JS que lo convierte en campo requerido, no mayor de 128 caracteres.
  + Masivo: Checkbox.
  + Activo: Checkbox.
  + Control asociado: Caja de texto, la cual muestra en forma dinámica todas las secciones y/o controles posibles mediante la comparación de lo escrito en el campo con la información almacenada en la base de datos.
  + Opción asociada: Caja de texto, está escondida por defecto, y aparece cuando es seleccionado un control asociado. muestra en forma dinámica las opciones que pertenecen al control seleccionado mediante la comparación de lo escrito en el campo con la información almacenada en la base de datos, y permite autocompletar mediante la selección de una opción.

Una vez enviado el formulario, se accede al método “saveControl” del modelo “Package”, el cual almacena un nuevo registro en la tabla “controls” de la base de datos SQLite asociada al paquete.

* Modificar: se crea un botón con una ruta al método “modifyControl” del controlador “inputEditionController”, el cual hace conexión con la base de datos SQLite del paquete y retorna la vista “modify\_control”, la cual contiene una ventana modal con un formulario con los mismos campos y validaciones utilizados en la vista “new\_control”. Una vez enviado el formulario, se accede al método “modifyControl” del modelo “Package”, el cual modifica el registro en la tabla “controls” de la base de datos SQLite asociada al paquete, mediante el ID del control.
* Información: se crea un botón con una ruta al método “infoControl” del controlador “inputEditionController”, el cual hace conexión con la base de datos SQLite del paquete y retorna la vista “info\_control”, la cual contiene una ventana modal con información de la pestaña, estructurada de la siguiente forma
  + Nombre
  + Tipo de control
  + Ayuda
  + Opción predeterminada
  + Palabra clave
  + Masivo
  + Activo
  + Control asociado
  + Opción asociada
* Eliminar: se crea un botón que acciona una ventana modal de confirmación con una ruta al método “deleteControl” del controlador “inputEditionController”, el cual utiliza el método “deleteControl” del modelo “Package” para eliminar el registro de la base de datos SQLite asociada al paquete.

1. Opciones: Se maneja el CRUD de la siguiente manera:

* Agregar: se crea un botón con una ruta al método “newOption” del controlador “inputEditionController”, el cual hace conexión con la base de datos SQLite del paquete y retorna la vista “new\_option”, la cual contiene una ventana modal con un formulario con los siguientes campos:
  + Nombre: Caja de texto, la cual tiene una validación JS que lo convierte en campo requerido, alfanumérico, de longitud no menor de 5 caracteres y no mayor de 128 caracteres
  + Valor: Caja de texto, la cual tiene una validación JS que lo convierte en campo requerido.
  + Palabra clave: Caja de texto, la cual tiene una validación JS que lo convierte en campo requerido, alfanumérico, no mayor de 128 caracteres.

Una vez enviado el formulario, se accede al método “saveOption” del modelo “Package”, el cual almacena un nuevo registro en la tabla “options” de la base de datos SQLite asociada al paquete.

* Modificar: se crea un botón con una ruta al método “modifyOption” del controlador “inputEditionController”, el cual hace conexión con la base de datos SQLite del paquete y retorna la vista “modify\_option”, la cual contiene una ventana modal con un formulario con los mismos campos y validaciones utilizados en la vista “new\_option”. Una vez enviado el formulario, se accede al método “modifyOption” del modelo “Package”, el cual modifica el registro en la tabla “options” de la base de datos SQLite asociada al paquete, mediante el ID de la opción.
* Información: se crea un botón con una ruta al método “infoOption” del controlador “inputEditionController”, el cual hace conexión con la base de datos SQLite del paquete y retorna la vista “info\_option”, la cual contiene una ventana modal con información de la pestaña, estructurada de la siguiente forma:
  + Nombre
  + Valor
  + Palabra Clave
* Eliminar: se crea un botón que acciona una ventana modal de confirmación con una ruta al método “deleteOption” del controlador “inputEditionController”, el cual utiliza el método “deleteOption” del modelo “Package” para eliminar el registro de la base de datos SQLite asociada al paquete.