

Decisión sobre los Bugs de la Plataforma Correplayas Herencia Proyecto DAW

A) Mi reflexión a los Bugs de la Plataforma Correplayas.

Decidí apostar por el desarrollo de mi aplicación web para el proyecto fin de ciclo por los lenguajes de base sin apoyo de frameworks, pues no sé habían visto en la profundidad adecuada como para usarlos en un proyecto.

Adopté los principios de una arquitectura MVC pero sin implementar un sistema de URL amigable porque pensé en un principio que me sería más fácil el desarrollo el uso de comandos pasados como variable url, que a fin de cuentas era un prototipo académico.

La creación del micro -framework o núcleo de la propia plataforma me consumió mucho tiempo, y tuve que recortar estratégicamente para llegar a los plazos de entrega. Entonces decidí presentar tres de los seis gestores. Los cuáles desarrollé y fue la primera versión que presenté.

Mientras programaba los gestores de esta primera versión, me fui dando cuenta que mi pseudo-MVC tenía carencias en el sistema por comandos y en el flujo de navegación. No obstante, no podía detenerme en resolverlo porque tenía el tiempo pisando me los talones y necesitaba una versión funcional que presentar.

Finalmente, por una serie de avatares el tutor me solicitó que completase el desarrollo completo de la aplicación con un margen de tiempo de sólo un par de semanas, contando con la documentación final y preparación de la defensa.

Entonces me puse a completar el gestor de participantes y programar a los gestores censos, aves y observatorios. Aquí fue donde las carencias del psuedo MVC fueron más latentes, pero no tenía tiempo para refactorizaciones en dicha línea. El principal bug el engorroso sistema de navegación que había implementado, que me hacía difícil el navegar en el gestor de censos, el más complejo de todos. No obstante, logré dejarlo todo programado a tiempo y en una segunda versión funcional.

La tercera versión funcional tiene el objetivo de adaptar esta aplicación web como recurso profesional para mi portafolio. En esta tarea procedo a corregir fallos menores y a implantar el modo demostración de forma que un usuario interesado pueda navegar como un administrador pero sin tener permiso de escritura en la base de datos.

La necesidad de implementar un Modo Demostración reveló de forma definitiva el fallo de flujo subyacente en el sistema de comandos. En ese momento, realicé un diagnóstico completo de la arquitectura y propuse la solución definitiva (refactorizar a un MVC estándar).

Sin embargo, mi Discernimiento me indicó que invertir tiempo en una refactorización total era una mala decisión de ingeniería, dado que las carencias del pseudo-MVC ya no justificaban su continuidad.

Por lo tanto, apliqué un parche quirúrgico de Lógica Pura para estabilizar el modo Demostración y cerré el ciclo. Mi foco se traslada ahora al estudio y uso de Laravel, un framework que me permitirá construir sistemas escalables y sin las carencias arquitectónicas del desarrollo original. Mi valor es saber cuándo arreglar y cuándo empezar de cero.

Decisión sobre los Bugs de la Plataforma Correplayas Herencia Proyecto DAW

B) Mi propuesta de solución en el enrutador: ¿Cómo procedo?

Llegado a este punto pienso que podría pasar el identificador en la URL junto al comando y otros datos auxiliares que se requieran. Esta información de contexto del gestor se puede almacenar en una variable de sesión, cuya estructura puede ser:

- `$_SESSION['contexto']['idPrincipal']` => ID principal de elemento seleccionado.
- `$_SESSION['contexto']['idAuxiliar1']` => ID auxiliar1 de elemento seleccionado.
- `$_SESSION['contexto']['idAuxiliar2']` => ID auxiliar2 de elemento seleccionado.
- `$_SESSION['contexto']['navegacion']` => Contiene pila de urls con flujo navegación.
- `$_SESSION['contexto']['modoAdmin']` => Bandera control del modo restringido.
- `$_SESSION['contexto']['modoUsuarios']` => Bandera control del modo usuarios.
- `$_SESSION['contexto']['modoCenso']` => Bandera control edición censo plataforma.
- `$_SESSION['contexto']['datos']` => Datos de formularios saneados a procesar.

Entonces el enrutador se encarga de procesar la URL de la forma siguiente cuando hay usuario logueado:

- Si recibo comando procedo a recuperarlo y almacenarlo en una variable normal y además guardo en el contexto del gestor el identificador de elemento si se recibe. Por último, inicializo la pila con la URL actual y proceso el comando recibido.
- Descompongo el \$comando recibido en \$gestor, \$acción y \$fase. Luego genero la \$orden compuesta por \$accion:\$fase que se enviara a los métodos por defecto como argumento de cada controlador del gestor o núcleo.
- Si recibo un identificador auxiliar de elemento lo guardo en el contexto del gestor.
- Si recibo el estado del modo restringido lo guardo en el contexto del gestor. De lo contrario, lo establezco por defecto al valor falso.
- Si recibo el estado del modo usuarios lo guardo en el contexto del gestor. De lo contrario, lo establezco por defecto al valor falso.
- Si recibo un comando en fase de procesar y está establecido el POST, lo recupero y sanitizo con `input_array_filters` para guardarlo en el contexto del gestor.
- Cuando proceso un comando que solicita la vista por defecto de un gestor, se llama al método por defecto del controlador asociado a dicho gestor. Este método decide si muestra un listado o histórico, una vista detalles, excepción si el rol no está autorizado a ejecutarlo o ejecuta la \$orden recibida como argumento de forma que llama a las vistas y procesa peticiones CRUD.
- Cuando no recibo comando elimino la variable de sesión que almacena el contexto del gestor.

Decisión sobre los Bugs de la Plataforma Correplayas Herencia Proyecto DAW

← B.2) **Mi propuesta en cada método de acción de los controladores de cada gestor.**

Se debe actuar de la siguiente forma:

← B.2.1) **Vista:** ¿Cómo procedo?

- Necesito comprobar aquí el permiso del usuario para ejecutar la vista.
- Al inicio recupero la URL actual de la pila.
- Compruebo si el identificador del elemento es nulo para sí procede recuperarlo de la sesión o lanzar excepción según proceda. De lo contrario, recupero dicho identificador directamente desde el contexto del gestor.
- Se pasa el identificador de elemento a la vista.
- Se pasa la última dirección URL en la pila (end) como enlace para el botón volver de la vista que devuelve al usuario a la vista anterior.
- Si se renderiza la vista y se guarda URL en pila.
- Si se lanza cualquier excepción (inclusive el modo lectura) se pasa como URL actual recuperada para el botón aceptar.

B.2.2) **Procesa:** ¿Cómo procedo?

- Necesito comprobar aquí el permiso del usuario para ejecutar la vista.
- Al inicio recupero la URL actual de la pila.
- El identificador(es) del elemento se recibe desde el formulario correspondiente como campo oculto.
- Necesito definir un array constante (config-inc.php) con las reglas de limpieza del filtro para el formulario que se recibe en el enrutador y se pasado como variables de contexto del gestor.
- Necesito cambiar la asociación de datos procesados desde el POST original al nuevo método que se recibe como variable del contexto del gestor.
- Si la acción se procesa correctamente entonces recupero la anterior URL para devolver a usuario a la vista anterior al pulsar en aceptar sobre el mensaje de información de éxito.
- Si se lanza cualquier excepción se pasa como URL actual recuperada para el botón aceptar de forma que devuelve al usuario a la vista previa al proceso de acción.
- **OBSERVACIONES:** Esta lógica general quizás se necesite adaptar en algunos métodos de acciones en el controlador de los gestores.

B.3) **Mi propuesta para cada una de la plantillas afectadas:** ¿Cómo procedo?

Se ven afectadas todas las plantillas donde exista un botón de volver (atrás, salir, etcétera) o botones de acciones auxiliares. A los primeros hay que actualizar su URL de destino y a los segundo convertirlos de botones de formularios a una simple URL de destino con el formato:

Decisión sobre los Bugs de la Plataforma Correplayas Herencia Proyecto DAW

Formato de URL de peticiones GET

Https://correplayas.xampp/backoffice.php?comando=?&id=?&aux=?&admin=?&usuarios=?

Donde la variable de la URL significan lo siguiente:

- **comando**: Peticion que se le solicita al enrutador para que la ejecute.
- **Id**: Identificador de elemento principal.
- **Aux1**: Identificador de elemento auxiliar1.
- **Aux2**: Identificador de elemento auxiliar2.
- **Admin**: Bandera de estado para habilitar o no el modo restringido de los gestores.
- **Usuarios**: Bandera de estado para habilitar o no el modo de los listado en participantes.

Recuerda que todos estas variables son opcionales y que si no se reciben, dependiendo del contexto de usuario que navegue (logueado o visitante) se muestra en su ausencia, bien la página de inicio de backoffice o el portal público.

Todas las plantillas de listado o históricos se ven afectadas los elementos de filas asociadas a tablas dinámicas donde los botones de acción deben migrarse de botón de formulario a una URL con el formato indicado arriba, según el contexto de su funcionalidad.

B.4) Mi propuesta de mejora del Modo Sólo Lectura: ¿Cómo procedo?

- Cuando se detecta en el método ***ejecutarSql*** del núcleo que el contexto de la petición es de sólo lectura, quiero emular que al lanzar esa excepción ponerle con código de error 1836 (***DB_READ_ONLY_MODE***) para que tenga coherencia, aunque realmente los permisos de la base de datos no sean de solo lectura.

Los procedimiento de procesamiento que lleven parejos peticiones SQL de escritura al igual que se controla la excepción por error de integridad en la base de datos debe controlar el error ***DB_READ_ONLY_MODE***.