

Lima, 19 de diciembre de 2022

**CARTA 003-WTS-2022**

Señor

JORGE LUIS TAVARA VALLEJOS

Ejecutivo de Tecnologías de la Información

Programa Nacional Plataformas de Acción para la Inclusión Social - PAIS

Presente. -

Por la presente me dirijo a usted, a fin de presentar el informe por el servicio para contratar el SERVICIO ESPECIALIZADO PARA DISEÑAR E IMPLEMENTAR LOS APLICATIVOS MÓVILES PARA EL MÓDULO DE SEGUIMIENTO Y MONITOREO DE LOS PROYECTOS TAMBOS EN EJECUCIÓN PARA LA ALTA DIRECCIÓN DEL PROGRAMA NACIONAL "PLATAFORMAS DE ACCIÓN PARA LA INCLUSIÓN SOCIAL" – PAIS. Único entregable, según lo indicado en los Términos de Referencia de las Orden de Servicio N.º 0002402-2022, que se emitió el día 06/12/2022.

Asimismo, se adjunta al presente los siguientes documentos:

- Un (01) Informe N° 003-2022-WTS, de fecha 19 de diciembre de 2022, indicando los servicios brindados.
- Un (01) Recibo de Honorarios – E001-XX.

Sin otro particular, aprovecho la oportunidad para expresarle las muestras de mi especial consideración y estima.

Atentamente,



---

**Wilmer Tucto Santos**  
**RUC: 10475322628**



**ÚNICO ENTREGABLE**

## INFORME N° 003-2022-WTS

PARA : JORGE LUIS TAVARA VALLEJOS  
Ejecutivo de la Unidad de Tecnologías de la Información  
Programa Nacional PAIS

DE : WILMER TUOTO SANTOS

ASUNTO : INFORME CON RELACION AL SERVICIO BRINDADO

REFERENCIA : ORDEN DE SERVICIO N° 0002402-2022

FECHA : 19/12/2022

---

Me dirijo a usted para saludarlo cordialmente y a la vez, informarle lo siguiente:

### I. ANTECEDENTES:

Mediante Decreto Supremo N° 013-2017-MIDIS, se establece el Programa Nacional “Plataformas de Acción para la Inclusión Social – PAIS” sobre la base del Programa Nacional Tambos, creado por Decreto Supremo N° 016-2013- VIVIENDA, el cual tiene por finalidad mejorar la calidad de vida de la población pobre y extremadamente pobre, especialmente la asentada en los centros poblados rurales o rural dispersa, coadyuvando a su desarrollo económico, social y productivo que contribuyan a su inclusión social. Asimismo, tiene como objetivo, permitir el acceso de la población pobre y extremadamente pobre, especialmente la asentada en los centros poblados del área rural y de manera dispersa, a los servicios y actividades, en materias sociales y productivas que brinda el Estado.

### II. ALCANCES Y DESCRIPCIÓN DEL SERVICIO

1. Levantar información y requerimientos del área usuaria para el análisis funcional del aplicativo móvil Multiplataforma (Android / IOS) que permita la visualización del estado de avances de las obras de proyectos en ejecución tambos a nivel nacional.
2. Evaluar el flujo del FrontEnd y Backend del aplicativo móvil Multiplataforma (Android / IOS) para visualizar el estado de los avances de todos los proyectos tambos a nivel nacional y el detalle de un proyecto tambo con sus imágenes de monitoreo para el Programa Nacional PAIS.
3. Analizar las consultas a base de datos de procedimientos almacenados para visualizar el estado de los avances de todos los proyectos, el detalle de un proyecto para el Programa Nacional PAIS.

### III. ENTREGABLE

1. Un (01) informe de los requerimientos funcionales para la implementación del aplicativo móvil Multiplataforma (Android / IOS) que permita la visualización del estado de los avances de todos los proyectos tambos a nivel nacional.
2. Un (01) informe de la implementación y flujo del FrontEnd y Backend del aplicativo móvil Multiplataforma (Android / IOS) para visualizar el detalle de un proyecto tambo y sus imágenes de monitoreo.



3. Un (01) informe con el análisis de la información generada a través de los Procedimientos Almacenados para visualizar el detalle de un proyecto tambo y sus imágenes de monitoreo.

Atentamente,



---

**WILMER TUCTO SANTOS**  
DNI N° 47532262



**1. informe de los requerimientos funcionales para la implementación del aplicativo móvil Multiplataforma (Android / IOS) que permita la visualización del estado de los avances de todos los proyectos tambos a nivel nacional**

El módulo de “seguimiento y monitoreo de los proyectos tambos” tiene como finalidad apoyar en la toma de decisiones para el personal administrativo y la alta dirección del programa Nacional “Plataforma de Acción para Inclusión Social – País” a través de indicadores e información en tiempo real acerca de los avances en los proyectos tambos a nivel nacional, es por ello que después de realizar el levantamiento de información, se han identificados los siguientes requerimientos funcionales:

<b>REQUERIMIENTO</b>	PAIS_UPS_RQ01: Mostrar el estado de avance de todos los proyectos tambos a nivel nacional
<b>ACTORES</b>	Personal Administrativo
<b>PRECONDICIÓN</b>	El usuario administrativo se autenticó con sus credenciales en el sistema
<b>PROPÓSITO</b>	
La aplicación mostrará un chart de tipo torta de la cantidad de proyectos tambo agrupados según el estado de su avance considerando las siguientes reglas: <ul style="list-style-type: none"><li>➤ <b>Culminado:</b> Cuando el estado de avance físico es igual al 100% y está representado de color azul</li><li>➤ <b>Alto:</b> Cuando el estado de avance físico está entre 50% y menor a 100% y está representado de color verde.</li><li>➤ <b>Medio:</b> Cuando el estado de avance físico está entre 30% y menor a 50% y está representado de color amarillo.</li><li>➤ <b>Bajo:</b> Cuando el estado de avance físico está entre 0% y menor a 30% y está representado de color rojo.</li></ul>	

<b>REQUERIMIENTO</b>	PAIS_UPS_RQ02: Filtrar los proyectos tambo al seleccionar cada parte del chart tipo torta.
<b>ACTORES</b>	Personal Administrativo
<b>PRECONDICIÓN</b>	El usuario administrativo se autenticó con sus credenciales en el sistema
<b>PROPÓSITO</b>	
La aplicación permitirá aplicar filtros por cada sector de chart de tipo torta en donde mostrara la lista de proyectos tambo de acuerdo al filtro aplicado, los filtros considerados son los siguientes: <ul style="list-style-type: none"><li>➤ <b>Culminado:</b> Cuando el estado de avance físico es igual al 100% y está representado de color azul.</li><li>➤ <b>Alto:</b> Cuando el estado de avance físico está entre 50% y menor a 100% y está representado de color verde.</li><li>➤ <b>Medio:</b> Cuando el estado de avance físico está entre 30% y menor a 50% y está representado de color amarillo.</li><li>➤ <b>Bajo:</b> Cuando el estado de avance físico está entre 0% y menor a 30% y está representado de color rojo.</li></ul>	

<b>REQUERIMIENTO</b>	PAIS_UPS_RQ03: Lista de todos los proyectos tambo a nivel nacional
<b>ACTORES</b>	Personal Administrativo



<b>PRECONDICIÓN</b>	El usuario administrativo se autentificó con sus credenciales en el sistema
<b>PROPÓSITO</b>	
<p>La aplicación mostrará todos los proyectos tambo a nivel nacional en contenedores bordeados por color según el porcentaje del avance físico que se consideró en los requerimientos anteriores, cada contenedor mostrara los siguientes datos del proyecto tambo:</p> <ul style="list-style-type: none"> <li>➤ Porcentaje del Avance Físico del Proyecto</li> <li>➤ Nombre Tambo</li> <li>➤ Estado de Proyecto</li> <li>➤ Departamento, Provincia y Distrito del proyecto</li> <li>➤ Fecha de Término Estimado del proyecto</li> <li>➤ Código Único Identificador del Proyecto</li> </ul> <p>Así mismo te permite visualizar el detalle del proyecto tambo al seleccionar el contenedor.</p>	

<b>REQUERIMIENTO</b>	PAIS_UPS_RQ04: Visualizar Imágenes de los monitores por proyecto tambo
<b>ACTORES</b>	Personal Administrativo
<b>PRECONDICIÓN</b>	El usuario administrativo se autentificó con sus credenciales en el sistema
<b>PROPÓSITO</b>	
<p>La aplicación mostrara las imágenes de monitoreo en un carrusel de acuerdo al código Único Identificador del proyecto, las imágenes del monitoreo que se visualizaran son los siguientes:</p> <ul style="list-style-type: none"> <li>➤ Imagen de Actividad de la Partida Ejecutada.</li> <li>➤ Imagen del Problema Identificado.</li> <li>➤ Imagen del riesgo Identificado.</li> </ul>	

<b>REQUERIMIENTO</b>	PAIS_UPS_RQ05: Visualizar detalle del proyecto Tambo
<b>ACTORES</b>	Personal Administrativo
<b>PRECONDICIÓN</b>	El usuario administrativo se autentificó con sus credenciales en el sistema
<b>PROPÓSITO</b>	
<p>La aplicación mostrará el detalle de proyecto previamente seleccionado de la lista de proyectos en donde te permite visualizar el siguiente detalle del proyecto:</p> <ul style="list-style-type: none"> <li>➤ Estado del Proyecto.</li> <li>➤ Departamento, Provincia y Distrito del Proyecto.</li> <li>➤ Código Único Identificador.</li> <li>➤ Porcentaje de Avance Físico del Proyecto.</li> <li>➤ Código de SNIP del Proyecto.</li> <li>➤ Sub Estado del Proyecto.</li> <li>➤ Estado de Saneamiento del Proyecto.</li> <li>➤ Modalidad de Contratación del Proyecto.</li> <li>➤ Fecha de Inicio del Proyecto.</li> <li>➤ Fecha de Término Estimado del Proyecto.</li> <li>➤ Monto de Inversión del Proyecto.</li> <li>➤ Costo Ejecutado Acumulado del Proyecto.</li> </ul>	

- Costo Estimado Final del Proyecto.
- Nombre del Residente.
- Nombre del Supervisor.
- Nombre del Coordinador Regional del Proyecto.
- Código del Residente.
- Código del Supervisor.
- Código del Coordinador Regional del Proyecto.
- Latitud de la Ubicación del Proyecto.
- Longitud de la Ubicación del Proyecto.



## 2. Informe de la implementación y flujo del FrontEnd y Backend del aplicativo móvil Multiplataforma (Android / IOS) para visualizar el detalle de un proyecto tambo y sus imágenes de monitoreo.

### 2.1 Descargar usuarios de la base de datos central a la base de datos del aplicativo mediante API Rest.

Al respecto, se desarrolló el repositorio y modelo en la capa de datos del backend del proyecto con la finalidad de conectarse a la fuente de datos externa mediante el api Rest **"listarUsuariosApp"**, dicha api devuelve una lista de usuarios que luego serán almacenadas en la base de datos local del dispositivo móvil.

De acuerdo con la documentación del proyecto, la descarga de usuarios desde la base de datos central a la base de datos local del aplicativo debe suceder de forma automática cuando se utilice el aplicativo por primera vez. En cumplimiento de esa funcionalidad, se ha desarrollado la lógica de negocio para descargar usuarios de forma automática cada vez que se inicie el aplicativo.

Cabe precisar que, para conectarse a la Api Rest, el dispositivo móvil debe contar con acceso a internet, en caso contrario el aplicativo lanzara un mensaje de alerta indicando al usuario que debe conectarse a internet.

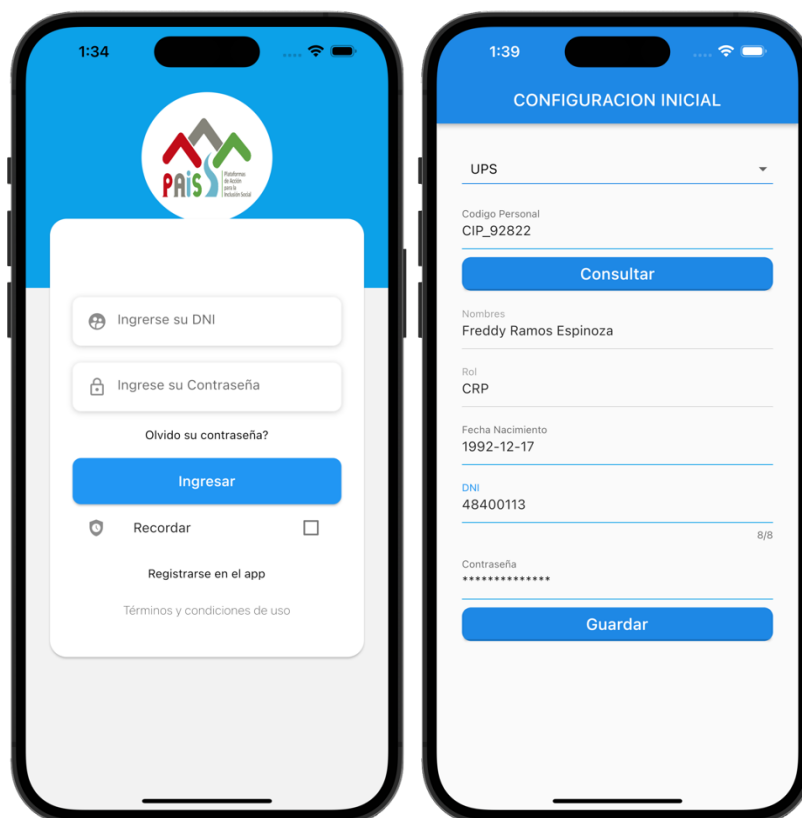


Imagen de inicio sesión y registro en el aplicativo móvil



```

1
2 /*
3 Obtiene los datos del usuario
4 @String codigo
5 @String clave (Opcional) Si clave es vacio solo obtiene la info del usuario por el codigo
6 */
7 Future<UserModel> getUserLogin(
8     String codigo,
9     String clave,
10 ) async {
11     UserModel oUserLogin = await Get.find<MainService>().getUserByCode(codigo);
12
13     if (clave != "") {
14         if (oUserLogin.clave == clave) {
15             return oUserLogin;
16         }
17     } else {
18         return oUserLogin;
19     }
20
21     return Future.error(
22         "Usuario y/o Clave incorrecto, vuelve a intentarlo mas tarde.",
23     );
24 }
25
26 /*
27 Inserta / Actualiza datos del Usuario
28 @UserModel o
29 */
30 Future<UserModel> insertUser(UserModel o) async {
31     try {
32         UserModel oUserLogin = await Get.find<MainService>().insertUserDb(o);
33         return oUserLogin;
34     } catch (oError) {
35         return Future.error(
36             oError.toString(),
37         );
38     }
39 }

```

Código fuente del modelo para descargar usuarios.

## 2.2 Ingreso a la opción de seguimiento y monitoreo de proyectos tambos.



Imagen de la opción/menú de SEGUIMIENTO Y MONITOREO

Conforme a lo señalado en la actividad, si el usuario ingresa a la opción/menú de SEGUIMIENTO Y MONITOREO de aplicativo, este mostrará un Gráfico estadístico circular (pie chart) y la lista de proyectos tambos.

### 2.3 Descargar los datos generales de los proyectos tambo desde base de datos central a la memoria del dispositivo móvil.

Conforme a lo señalado en la actividad, se desarrolló el repositorio y modelo en la capa de datos del proyecto, con el fin de conectarse a la fuente de datos externa mediante el api “**listarTramaProyecto**”, dicha api devuelve una lista de proyectos, las cuales son almacenadas en la memoria del dispositivo móvil.

Cada vez que se ingrese a la opción/menú de SEGUIMIENTO Y MONITOREO del aplicativo móvil y siempre en cuando tenga acceso a internet, este podrá obtener los registros de proyectos que se hayan registrado en la base de datos central. Adicionalmente, el aplicativo móvil cuenta con una opción para refrescar la lista de proyectos de forma manual a través de un botón sin la necesidad de volver a ingresar a la opción/menú de SEGUIMIENTO Y MONITOREO.

Cabe precisar que, para conectarse a la Api Rest, el celular debe contar con acceso a internet, en caso contrario el aplicativo lanzara un mensaje de alerta indicando al usuario que debe conectarse a internet.

```
1
2  /*
3  Obtiene la lista de proyectos segun el ROL del Usuario y busqueda según la coincidencia
4  @UserModel o
5  @String search
6  */
7  Future<List<TramaProyectoModel>> getAllProyectoDashboard(
8      String search,
9  ) async {
10     return await Get.find<MainService>().getAllProyectoDashboard(search);
11 }
```

```
1
2  Future<HttpResponse<List<TramaProyectoModel>>> listarTramaProyecto() async {
3      return await _http.request<List<TramaProyectoModel>>>(
4          '${basePathApp}listarTramaproyecto',
5          method: "GET",
6          parser: (data) {
7              return (data as List)
8                  .map((e) => TramaProyectoModel.fromJson(e))
9                  .toList();
10          },
11      );
12 }
```

```

1
2 Future<HttpResponse<T>> request<T>({
3   String path, {
4     String method = "GET",
5     Map<String, dynamic>? queryParameters,
6     Map<String, dynamic>? data,
7     Map<String, dynamic>? formData,
8     Map<String, String>? headers,
9     T Function(dynamic data)? parser,
10  }) async {
11    try {
12      final response = await _dio.request(
13        path,
14        options: Options(
15          method: method,
16          headers: headers,
17        ),
18        queryParameters: queryParameters,
19        data: formData != null ? FormData.fromMap(formData) : data,
20      );
21
22      var oData;
23      try {
24        oData = response.data["response"];
25      } catch (oError) {
26        oData = jsonDecode(response.data);
27      }
28
29      if (parser != null) {
30        return HttpResponse.success<T>(parser(oData));
31      }
32      return HttpResponse.success<T>(oData);
33    } catch (oError) {
34      _log.e(
35        oError.toString(),
36      );
37
38      int statusCode = 0;
39      String message = "Unknown error";
40      dynamic data;
41      if (oError is DioError) {
42        statusCode = -1;
43        message = oError.message;
44        if (oError.response != null) {
45          statusCode = oError.response!.statusCode!;
46          if (oError.response!.statusMessage! != "") {
47            message = oError.response!.statusMessage!;
48          }
49          data = oError.response!.data;
50        }
51      }
52
53      return HttpResponse.fail<T>({
54        statusCode: statusCode,
55        message: message,
56        data: data,
57      });
58    }
59  }

```

Código fuente del modelo de descarga de datos generales de proyectos tambos.

## 2.4 Desarrollar interfaz para visualizar los datos generales de los proyectos tambos.

Conforme a lo señalado en la actividad, se desarrolló el código de la interfaz en la capa de presentación del proyecto para visualizar la lista de proyectos y mostrara un Gráfico estadístico circular (pie chart).

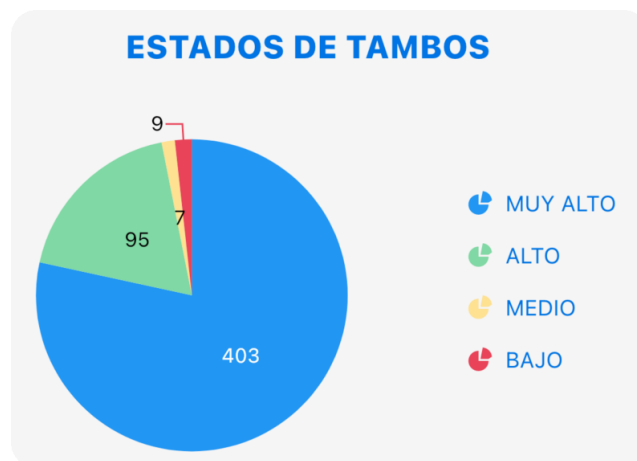
El gráfico estadístico circular está definido de acuerdo con el porcentaje de avance físico del proyecto y conformado por las siguientes secciones:

**MUY ALTO:** Esta sección tiene definido el color AZUL y está determinado por aquellos proyectos cuyo avance físico de proyecto esté en el rango de 100% como valor.

**ALTO:** Esta sección tiene definido el color VERDE y está determinado por aquellos proyectos cuyo avance físico de proyecto esté en el rango de mayor/igual a 50% al menor a 100% como valor.

**MEDIO:** Esta sección tiene definido el color AMARILLO y está determinado por aquellos proyectos cuyo avance físico de proyecto esté en el rango de mayor a 30% al menor a 50% como valor.

**BAJO:** Esta sección tiene definido el color ROJO y está determinado por aquellos proyectos cuyo avance físico de proyecto esté en el rango de menor/igual a 0% al menor/igual a 30% como valor.



```
1
2 Color getColorAvancefisico(dynamic oProyecto) {
3   try {
4     return ((double.parse(oProyecto.avanceFisico.toString()) * 100) == 100
5       ? Colors.blue
6       : (double.parse(oProyecto.avanceFisico.toString()) * 100) >= 50
7       ? Colors.green
8       : (double.parse(oProyecto.avanceFisico.toString()) * 100) <= 30
9       ? Colors.red
10      : Colors.yellow);
11   } catch (oError) {
12     return Colors.black;
13   }
14 }
```

```
1
2 int getAvancefisicoChar(dynamic oProyecto) {
3   try {
4     return ((double.parse(oProyecto.avanceFisico.toString()) * 100) == 100
5       ? 1 /* MUY ALTO*/
6       : (double.parse(oProyecto.avanceFisico.toString()) * 100) >= 50
7       ? 2 /* ALTO*/
8       : (double.parse(oProyecto.avanceFisico.toString()) * 100) <= 30
9       ? 4 /* BAJO*/
10      : 3); /* MEDIO */
11   } catch (oError) {
12     return 4;
13   }
14 }
```

Código fuente del modelo de distribución del grafico estadístico según avance físico de los proyectos tambos.

```

1
2 Future<void> buildData() async {
3   aProyecto1 = [];
4   aProyecto2 = [];
5   aProyecto3 = [];
6   aProyecto4 = [];
7
8   aProyectoAll = await mainController.getAllProyectoDashboard("");
9
10  aProyectoAll.sort((a, b) => a.avanceFisico.compareTo(b.avanceFisico));
11
12  for (var o in aProyectoAll) {
13    int iSection = getAvanceFisicoChar(o);
14    switch (iSection) {
15      case 1: /* MUY ALTO */
16        aProyecto1.add(o);
17        break;
18      case 2: /* ALTO */
19        aProyecto2.add(o);
20        break;
21      case 3: /* BAJO */
22        aProyecto3.add(o);
23        break;
24      case 4: /* MEDIO */
25        aProyecto4.add(o);
26        break;
27      default:
28    }
29  }
30
31  setState(() {
32    aProyecto = aProyectoAll;
33    chartData = [
34      //1 - Verde      8808A4
35      //2 - Amarillo   FEE191
36      //3 - Rojo       E84258
37      ChartData('MUY ALTO', aProyecto1.length, const Color(0xFF2196F3)),
38      ChartData('ALTO', aProyecto2.length, const Color(0xFF8808A4)),
39      ChartData('MEDIO', aProyecto3.length, const Color(0xFFFFE191)),
40      ChartData('BAJO', aProyecto4.length, const Color(0xFFE84258)),
41    ];
42  });
43 }
44
45 Color getColorAvanceFisico(dynamic aProyecto) {
46   try {
47     return ((double.parse(aProyecto.avanceFisico.toString()) * 100) == 100
48       ? Colors.blue
49       : (double.parse(aProyecto.avanceFisico.toString()) * 100) >= 50
50       ? Colors.green
51       : (double.parse(aProyecto.avanceFisico.toString()) * 100) <= 30
52       ? Colors.red
53       : Colors.yellow);
54   } catch (e) {
55     return Colors.black;
56   }
57 }
58
59 int getAvanceFisicoChar(dynamic aProyecto) {
60   try {
61     return ((double.parse(aProyecto.avanceFisico.toString()) * 100) == 100
62       ? 1 /* MUY ALTO */
63       : (double.parse(aProyecto.avanceFisico.toString()) * 100) >= 50
64       ? 2 /* ALTO */
65       : (double.parse(aProyecto.avanceFisico.toString()) * 100) <= 30
66       ? 4 /* BAJO */
67       : 3); /* MEDIO */
68   } catch (e) {
69     return 4;
70   }
71 }
72
73 double getAvanceFisico(dynamic aProyecto) {
74   try {
75     return double.parse(aProyecto.avanceFisico.toString());
76   } catch (e) {
77     return 1;
78   }
79 }
80
81 String getAvanceFisicoText(dynamic aProyecto) {
82   try {
83     return "${(double.parse(aProyecto.avanceFisico.toString()) * 100).toStringAsFixed(2)).toString()}%";
84   } catch (e) {
85     return "N/A";
86   }
87 }

```

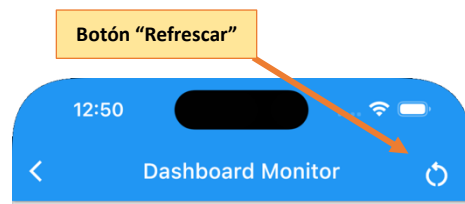
Código fuente del modelo de distribución del grafico estadístico según avance físico de los proyectos tambos.



Interfaz para visualizar la lista de proyectos y el gráfico circular

Conforme a lo señalado en la actividad, se desarrolló el código de la interfaz en la capa de presentación del proyecto para visualizar la lista de proyectos y mostrara un Gráfico estadístico circular (pie chart), La interfaz contara con los siguientes controles:

- **Refresh:** la interfaz contará con un botón refrescar en la parte superior con un icono de una flecha circular, en la cual al hacer click, volverá a consultar y obtener los registros de proyectos tambos a la base de datos central y volver a procesarlos para poder mostrar la lista actualizada en el dispositivo móvil. Detalle descrito en el punto 2.2.3 del documento.



- **Listar proyectos por cada categoría del gráfico estadístico** los proyectos se mostrarán en forma recuadros, con el nombre del proyecto como título, estado del proyecto, código CUI, indicador del avance del proyecto en barra de progreso circular.

Esta lista de proyectos se mostrará según el rango de avance físico determinado en el gráfico estadístico circular.

Detalle descrito en el punto 2.2.4 de documento.

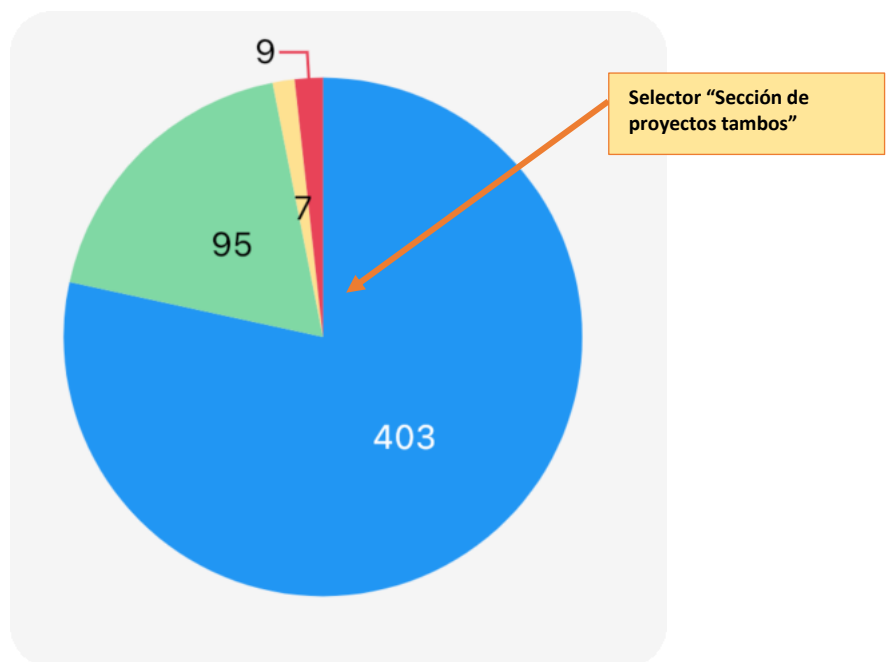


Imagen de los selectores y secciones del gráfico estadístico circular de proyectos tambos.

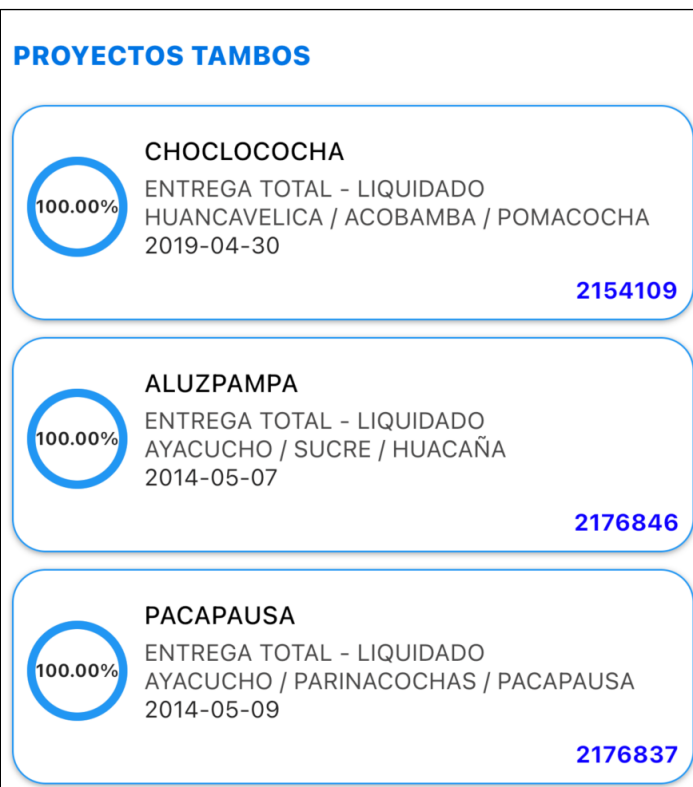


Imagen de resultados al hacer click en los selectores del gráfico estadístico circular.

## 2.5 Desarrollar interfaz para visualizar el detalle del proyecto seleccionado de la lista de proyectos y visualización de imágenes de monitoreo.

Conforme a lo establecido en la orden de servicio, se desarrolló la interfaz para visualizar el detalle del proyecto seleccionado e imágenes de monitoreos.

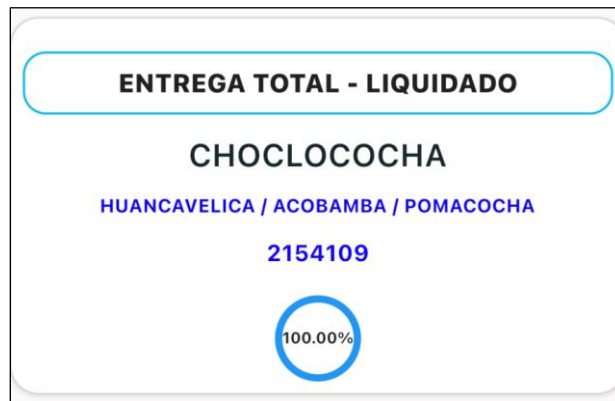
Al respecto, se desarrolló el código de la interfaz dentro de la capa de presentación del proyecto para mostrar el detalle de un proyecto seleccionado previamente de la interfaz de lista de proyectos. Como se pudo apreciar en la imagen adjunta, se hizo la prueba de la interfaz en un celular iphone, donde se muestran todos los datos de un proyecto, la interfaz contara con las siguientes secciones:

Listado de imagen de monitoreos tambo en una estructura de tipo carrousel.





Resumen estadístico y ubicación del proyecto seleccionado.



Detalle general del proyecto seleccionado.

<b>CÓDIGO DE SNIP</b>
209727
<b>SUB ESTADO DEL PROYECTO</b>
ENTREGA TOTAL
<b>ESTADO DE SANEAMIENTO DEL PROYECTO</b>
SI
<b>MODALIDAD DE <u>CONTRATACIÓN</u> DEL PROYECTO</b>

```
1
2  /*
3   Obtiene la detalle de proyectos segun  busqueda
4   @String search
5   */
6  Future<List<TramaProyectoModel>> getDetailProyectoDashboard(
7    String search,
8  ) async {
9    return await Get.find<MainService>().getAllProyectoDashboard(search);
10 }
```

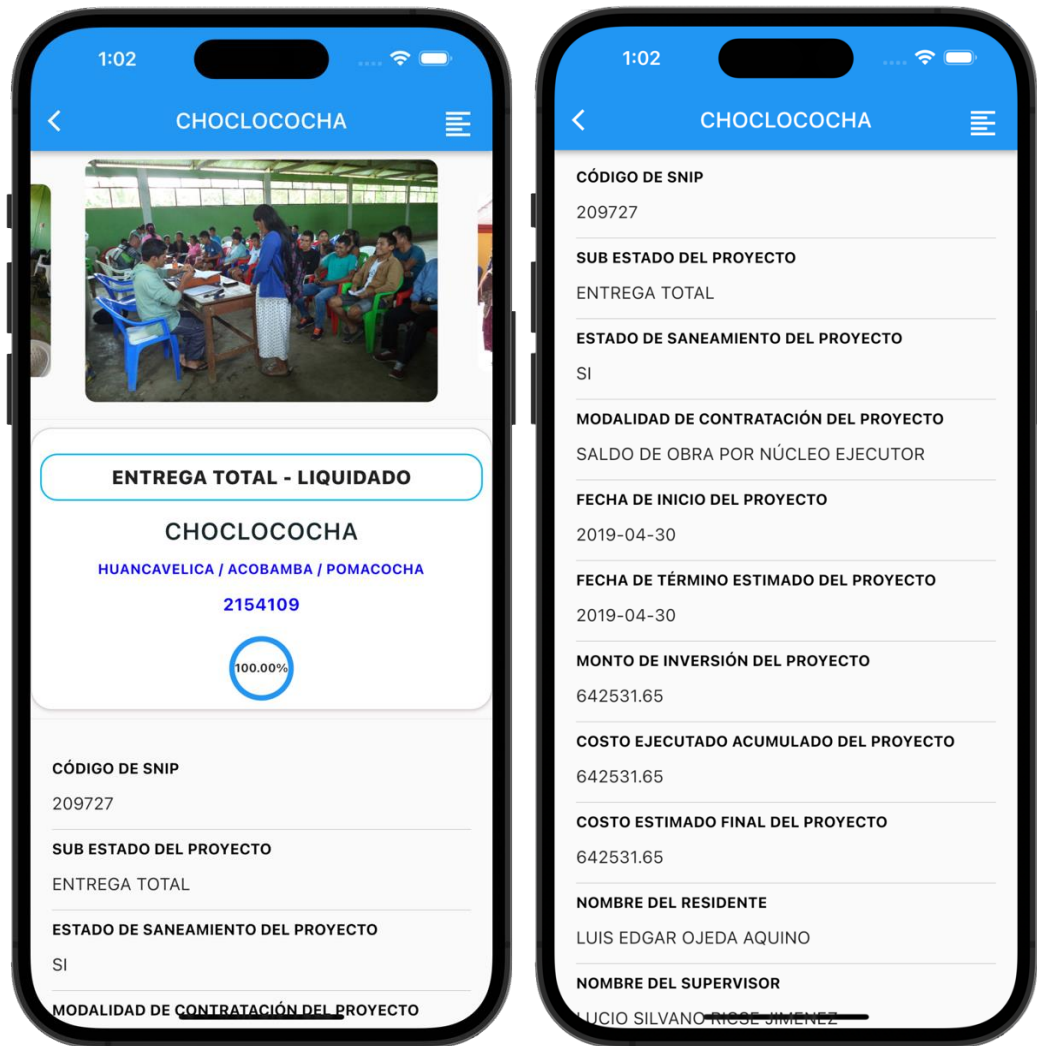
Código fuente del modelo de consulta de detalle de un proyecto tambos.

```

1
2 Future<HttpResponse<T>> request<T>(  
3   String path, {  
4     String method = "GET",  
5     Map<String, dynamic>? queryParameters,  
6     Map<String, dynamic>? data,  
7     Map<String, dynamic>? formData,  
8     Map<String, String>? headers,  
9     T Function(dynamic data)? parser,  
10  }) async {  
11    try {  
12      final response = await _dio.request(  
13        path,  
14        options: Options(  
15          method: method,  
16          headers: headers,  
17        ),  
18        queryParameters: queryParameters,  
19        data: formData != null ? FormData.fromMap(formData) : data,  
20      );  
21  
22      var oData;  
23      try {  
24        oData = response.data["response"];  
25      } catch (oError) {  
26        oData = jsonDecode(response.data);  
27      }  
28  
29      if (parser != null) {  
30        return HttpResponse.success<T>(parser(oData));  
31      }  
32      return HttpResponse.success<T>(oData);  
33    } catch (oError) {  
34      _log.e(  
35        oError.toString(),  
36      );  
37  
38      int statusCode = 0;  
39      String message = "Unknown error";  
40      dynamic data;  
41      if (oError is DioError) {  
42        statusCode = -1;  
43        message = oError.message;  
44        if (oError.response != null) {  
45          statusCode = oError.response!.statusCode!;  
46          if (oError.response!.statusMessage! != "") {  
47            message = oError.response!.statusMessage!;  
48          }  
49          data = oError.response!.data;  
50        }  
51      }  
52  
53      return HttpResponse.fail<T>(  
54        statusCode: statusCode,  
55        message: message,  
56        data: data,  
57      );  
58    }  
59  }

```

Código fuente del modelo de consulta de datos generales de proyectos tambos.



Interfaz para ver detalle de un proyecto

## 2.6 Arquitectura: Implementación y flujo del FrontEnd y Backend del aplicativo móvil.

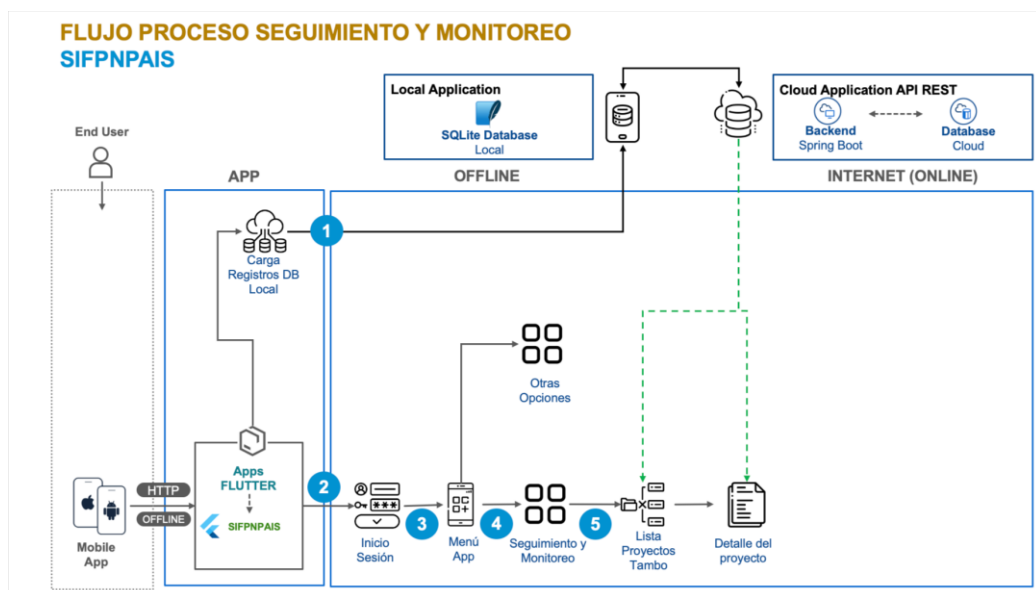


Imagen del flujo de proceso del proyecto.

*[Handwritten signature]*

## 2.7 Integración con las APIS para extraer data de la base de datos central hacia la base de datos del dispositivo móvil.

Como se mencionó párrafos arriba, el código fuente del proyecto se ha estructurado en capas, con la finalidad de facilitar su mantenimiento y desarrollo de mejoras del aplicativo móvil. En ese sentido, dentro de la arquitectura del proyecto, se ha creado una capa de datos destinada a alojar repositorios y modelos que se utilizaran para conectarse a las APIS externas, extraer la data y almacenarlos en la base de datos local del celular.

La capa de datos del proyecto está estructurada de la siguiente manera:

Carpeta	Descripción
Model	Paquete model para las representaciones de nuestro modelo de datos (entidades).
Repository	Paquete repository para las clases que establecen la comunicación con la base de datos y la API REST.
Service	Paquete service para agregar las clases que respondan a la funcionalidad y lógica de negocio.
Controller	Paquete controller para todos los endpoints que tenga nuestra aplicación, estable la comunicación y validación con la capa de presentación.
Database	Paquete de Base de Datos en la cual se genera las DDL (Lenguaje de Definición de Datos) y DML (Lenguaje de Manipulación de Datos) de las entidades (SELECT, INSERT, UPDATE, DELETE).
Api	Paquete de conexión y generación de Querys del servicio API REST.

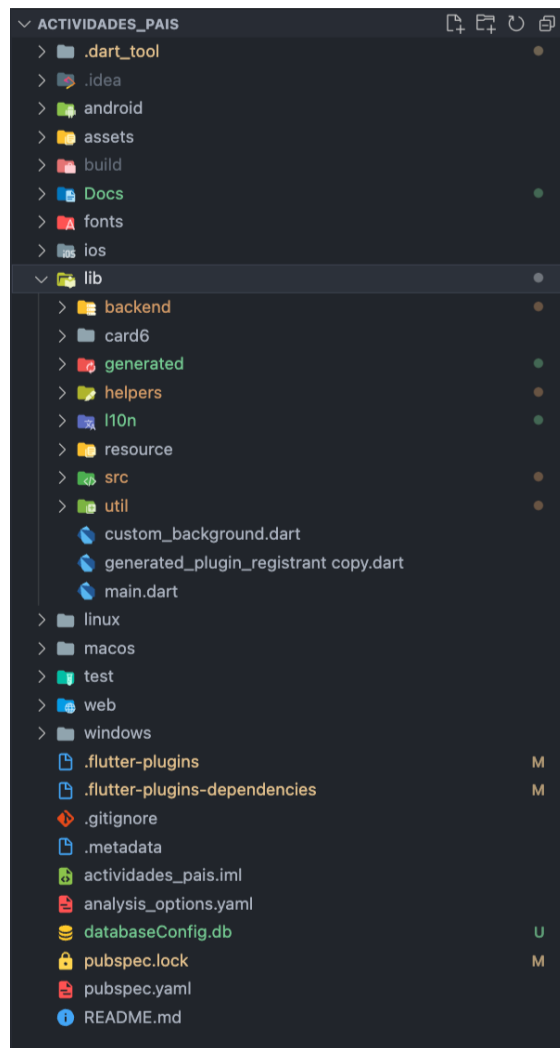


Imagen de la estructura general del proyecto.

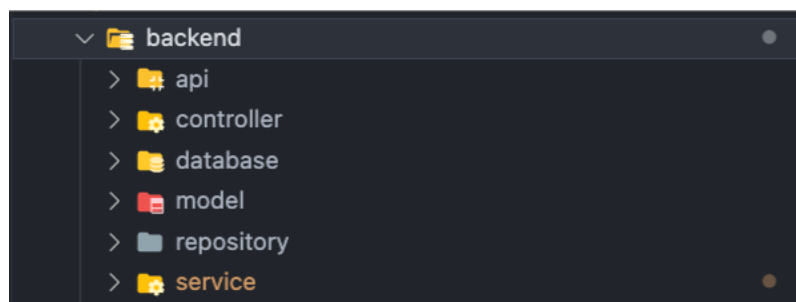


Imagen de la estructura backend del proyecto según definición de la arquitectura presentada.

**3. Informe con el análisis de la información generada a través de los procedimientos almacenados para visualizar el detalle de un proyecto tambo y sus imágenes de monitoreo**

El Sr. Javier Guillen, especialista en base de datos de la Unidad de Tecnologías de la Información, desarrollo un procedimiento almacenado denominado **“listarTramaMonitoreoMovilPaginado”** en la base de datos de monitoreos, dicho procedimiento almacenado se utilizará para obtener datos e imágenes de monitoreos tanto para la plataforma web y el aplicativo móvil.

El procedimiento almacenado **“listarTramaMonitoreoMovilPaginado”**, tiene la siguiente estructura:

Parámetros de Entradas:	
Nombre:	Descripción
P_SNIP_TAMBO	Código SNIP del proyecto tambo (opcional).
P_FEC_INICIO	Rango de fechas para realizar búsquedas de monitoreos (opcional).
P_FEC_FINAL	
P_ID_ESTADO_MONITOREO	Código identificador del estado de monitoreos (opcional).
P_PAGINA	Número índice de página
P_PAGESIZE	Tamaño de página

Respuesta del procedimiento almacenado	
Nombre:	Descripción
ITEM	Número de ítem
ID_MONITOREO	Código identificador del monitoreo
ESTADO_MONITOREO	Texto del estado del monitoreo
ID_USUARIO	Código identificador del usuario
USUARIO	Nombre del usuario
ROL	Rol del usuario
TAMBO	Nombre del proyecto tambo
SNIP	Código SNIP del proyecto tambo.
CUI	Código único identificador del proyecto
FECHA_MONITOREO	Fecha de registro de monitoreo
AVANCE_FISICO_ACUMULADO	Porcentaje de avance físico acumulado
ESTADO_AVANCE	Texto del estado de avance
OBSERVACIONES	Observaciones registradas en el monitoreo
FID_PBMA_IDTFCDO	Código identificador del problema identificado
PROBLEMA_IDENTIFICADO	Texto del problema identificado
NOM_ESTADO	Nombre del estado
IMG_PROBLEMA_1	Primera imagen del problema registrada en el monitoreo
IMG_PROBLEMA_2	Segunda imagen del problema registrada en el monitoreo
IMG_PROBLEMA_3	Tercera imagen del problema registrada en el monitoreo

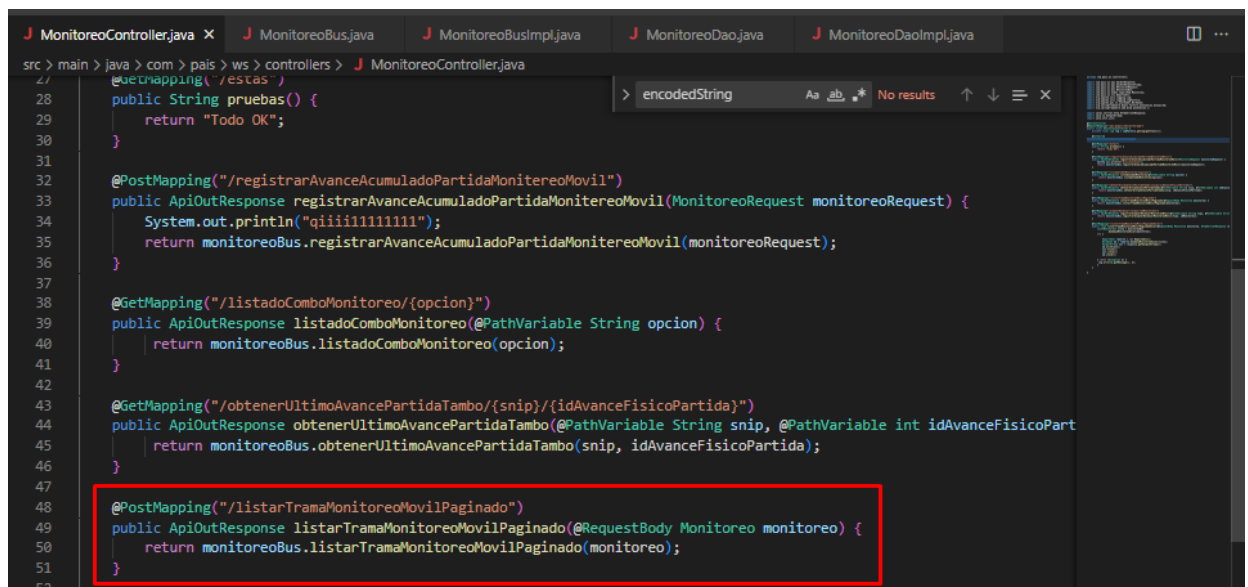


FID_ATNTVA_SLCION	Código identificador de la alternativa de solución
ALTERNATIVA_SOLUCION	Texto de la alternativa de solución
FID_RGO_IDTFCD0	Código identificador del riesgo identificado
RIESGO_IDENTIFICADO	Texto del riesgo identificado
IMG_RIESGO_1	Primera imagen del riesgo identificado en el monitoreo
IMG_RIESGO_2	Segunda imagen del riesgo identificado en el monitoreo
IMG_RIESGO_3	Tercera imagen del riesgo identificado en el monitoreo
FECHA_TERMINO_ESTIMADO	Fecha de finalización estimada del proyecto
LATITUD	Coordenada de latitud registrada en el monitoreo
LONGITUD	Coordenada de longitud registrada en el monitoreo
IMG_ACTIVIDAD_1	Primera imagen de la actividad
IMG_ACTIVIDAD_2	Segunda imagen de la actividad
IMG_ACTIVIDAD_3	Tercera imagen de la actividad

El procedimiento almacenado devuelve todos los datos de los monitoreos, así como las imágenes de los problemas, actividades y riesgos identificados.

### **API para consultar datos de un proyecto tambo y las imágenes de sus monitoreos**

Conforme a la estructura del procedimiento almacenado, se desarrolló un método POST llamado “**listarTramaMonitoreoMovilPaginado**” dentro del código de los servicios REST de monitoreos; específicamente en el archivo “**MonitoreoController.java**”.



```

src > main > java > com > pais > ws > controllers > J MonitoreoController.java
47
48 @PostMapping("/listarTramaMonitoreoMovilPaginado")
49 public ApiResponse listarTramaMonitoreoMovilPaginado(@RequestBody Monitoreo monitoreo) {
50     return monitoreoBus.listarTramaMonitoreoMovilPaginado(monitoreo);
51 }
52

```

Cada vez que el método POST reciba una petición de consulta, se debe enviar los siguientes parámetros en formato JSON:



```
{ "snip": "",  
  "fechaFin": "",  
  "fechaInicio": "",  
  "idEstadoMonitoreo": 0,  
  "pageIndex": 1,  
  "pageSize": 10  
}
```

Posteriormente, el método POST enviará los parámetros recepcionados a la clase **listarTramaMonitoreoMovilPaginado** de la interfaz **"MonitoreoBus"**.

```
public interface MonitoreoBus {  
  
    ApiOutResponse registrarAvanceAcumuladoPartidaMonitoreoMovil(MonitoreoRequest monitoreoRequest);  
    ApiOutResponse listadoComboMonitoreo(String opcion);  
    ApiOutResponse obtenerUltimoAvancePartidaTambo(String snip, int idAvanceFisicoPartida);  
    ApiOutResponse listarTramaMonitoreoMovilPaginado(Monitoreo monitoreo);  
    ApiOutResponse registrarAceptarRechazarMonitoreoMovil(String tipo, String idMonitoreo);  
  
    List<Monitoreo> bandejaMonitoreoExcel(Monitoreo monitoreo);  
}
```

Luego, la clase **listarTramaMonitoreoMovilPaginado** llama al método **"listarTramaMonitoreoMovilPaginado"** de la clase Service **"MonitoreoBusImpl"**.

```
MonitoreoBusImpl.java - código - Visual Studio Code  
J MonitoreoController.java J MonitoreoBus.java J MonitoreoBusImpl.java X J MonitoreoDao.java J MonitoreoDaoImpl.java  
src > main > java > com > pais > ws > busimpl > J MonitoreoBusImpl.java  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
@Override  
public ApiOutResponse listadoComboMonitoreo(String opcion) {  
    return monitoreoDao.listadoComboMonitoreo(opcion);  
}  
  
@Override  
public ApiOutResponse obtenerUltimoAvancePartidaTambo(String snip, int idAvanceFisicoPartida) {  
    return monitoreoDao.obtenerUltimoAvancePartidaTambo(snip, idAvanceFisicoPartida);  
}  
  
@Override  
public ApiOutResponse listarTramaMonitoreoMovilPaginado(Monitoreo monitoreo) {  
    return monitoreoDao.listarTramaMonitoreoMovilPaginado(monitoreo);  
}  
  
@Override  
public ApiOutResponse registrarAceptarRechazarMonitoreoMovil(String tipo, String idMonitoreo) {
```

Luego, la clase Service **MonitoreoBusImpl** llama a la clase **"listarTramaMonitoreoMovilPaginado"** de la Interfaz **"MonitoreoDao"**.





```

1 package com.pais.ws.dao;
2
3 import com.pais.ws.api.ApiOutResponse;
4 import com.pais.ws.api.ApiOutResponseTrama;
5 import com.pais.ws.model.seguridad.Monitoreo;
6
7 import java.util.List;
8
9 public interface MonitoreoDao {
10     ApiOutResponse registrarAvanceAcumuladoPartidaMonitoreoMovil(Monitoreo monitoreo);
11     ApiOutResponse listadoComboMonitoreo(String opcion);
12     ApiOutResponse obtenerUltimoAvancePartidaTambo(String snip, int idAvanceFisicoPartida);
13     ApiOutResponse listarTramaMonitoreoMovilPaginado(Monitoreo monitoreo);
14
15     ApiOutResponse registrarAceptarRechazarMonitoreoMovil(String tipo, String idMonitoreo);
16
17     List<Monitoreo> bandejaMonitoreoExcel(Monitoreo monitoreo);
18 }
19

```

Finalmente, la interfaz “**MonitoreoDao**” llama al servicio “**MonitoreoDaoImpl**” en la cual ejecuta el método “**listarTramaMonitoreoMovilPaginado**” para obtener los datos de los proyectos tambos e imágenes de los monitoreos de acuerdo a los parámetros enviados.

Como se puede apreciar en la imagen del código, se envía 5 parámetros al procedimiento almacenado **listarTramaMonitoreoMovilPaginado**.

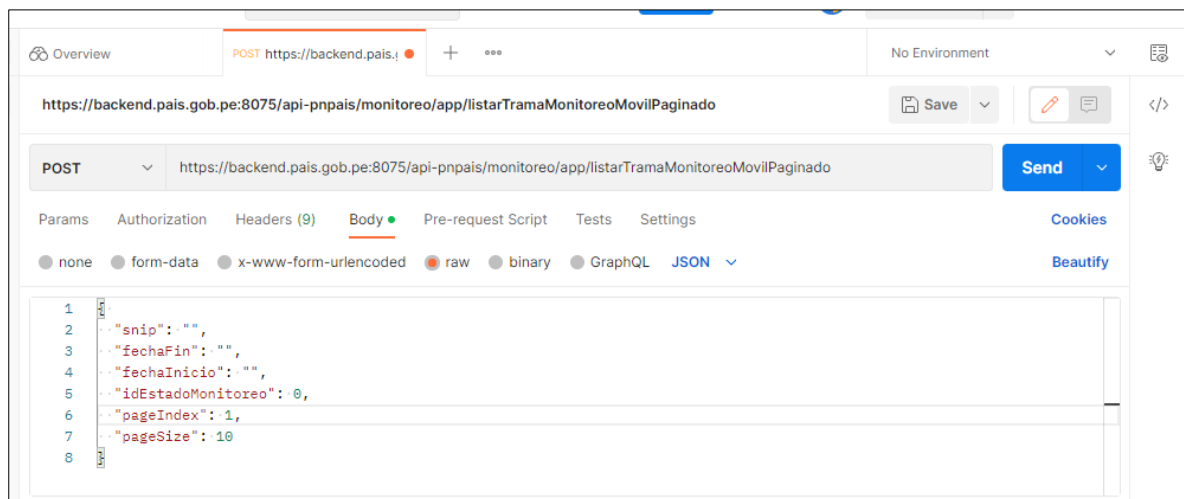
```

72
73 @Override
74 public ApiOutResponse listarTramaMonitoreoMovilPaginado(Monitoreo monitoreo) {
75     ApiOutResponse outResponse = new ApiOutResponse();
76     String mensajeRpta = "";
77     String[][] parametros = {
78
79         { "P_SNIP_TAMBO", monitoreo.getSnip() },
80         { "P_FEC_INICIO", monitoreo.getFechaFin() },
81         { "P_FEC_FINAL", monitoreo.getFechaFin() },
82         { "P_ID_ESTADO_MONITOREO", String.valueOf(monitoreo.getIdEstadoMonitoreo()) },
83         { "P_PAGINA", String.valueOf(monitoreo.getPageIndex()) },
84         { "P_PAGESIZE", String.valueOf(monitoreo.getPageSize()) },
85     };
86
87     bdr.ExecProcedureMessage(jdbcTemplate, jdbcCall, "MOVIL", "listarTramaMonitoreoMovilPaginado", parametros);
88     List<Map<String, Object>> rs = bdr.getResult();
89     mensajeRpta = bdr.getMensaje();
90     if (rs != null) {
91         if (rs.size() > 0) {
92             List<Monitoreo> lista = new ArrayList<Monitoreo>();
93             for (Map<String, Object> map : rs) {
94                 Monitoreo a = new Monitoreo();
95                 a.setItem(bdr.PString(map.get("ITEM")));
96                 a.setIdMonitoreo((map.get("ID_MONITOREO") == null) ? "" : map.get("ID_MONITOREO").toString());
97                 a.setNomEstado((map.get("ESTADO_MONITOREO") == null) ? "" : map.get("ESTADO_MONITOREO").toString());
98                 a.setNomEstado((map.get("ESTADO_MONITOREO") == null) ? "" : map.get("ESTADO_MONITOREO").toString());
99                 a.setIdUsuario(bdr.PLong(map.get("ID_USUARIO")));
100                 a.setUsuario(bdr.PString(map.get("USUARIO")));
101                 a.setRol(bdr.PString(map.get("ROL")));
102                 a.setTambo(bdr.PString(map.get("TAMBO")));
103
104                 a.setSnip(bdr.PString(map.get("SNIP")));
105                 a.setCui(bdr.PString(map.get("CUI")));
106                 a.setFechaMonitoreo(bdr.PString(map.get("FECHA_MONITOREO")));
107                 a.setAvanceFisicoAcumulado(bdr.PDouble(map.get("AVANCE_FISICO_ACUMULADO")));
108

```

La API para consultar los datos de los proyectos tambos y las imágenes de sus monitoreos se encuentra publicada mediante el siguiente enlace:

<https://backend.pais.gob.pe:8075/api-npnais/monitoreo/app/listarTramaMonitoreoMovilPaginado>



Consulta de la API con POSTMAN.

Formato JSON de la respuesta de la API:

```

{
  "codResultado": null,
  "msgResultado": null,
  "total": 2,
  "response": [
    {
      "item": "1",
      "idMonitoreo": "2153311_152549_2022-12-07",
      "idEstadoMonitoreo": 0,
      "estadoMonitoreo": null,
      "idUsuario": 3026,
      "usuario": "302",
      "idRol": 0,
      "rol": "0",
      "tambo": "COPALLIN DE ARAMANGO",
      "snip": "259323",
      "cui": "2178298",
      "fechaMonitoreo": "22-11-2022",
      "avanceFisicoAcumulado": 1,
      "idEstadoAvance": 0,
      "actividadPartidaEjecutada": null,
      "avanceFisicoPartida": 0,
      "observaciones": "obs",
      "imgActividad1": null,
      "imgActividad2": null,
      "imgActividad3": null,
      "problemaIdentificado": "CALCULO INEXACTO EN DURACIÓN DE LAS TAREAS",
      "idProblemaIdentificado": 1,
      "imgProblema1": "/9j/4AAQSkZJRgABAQEAYABgAAD/4SK+RXhpZgAATU0AKgAAAQgABgALAAIAAAmAAAB",
      "imgProblema2": "/9j/4bFwRXhpZgAASUkqAAgAAAAAABBAABAAAAABIAAAEBBAABAAAAgA0AAA8BAGAA",
      "imgProblema3": null,
    }
  ]
}

```

```
"alternativaSolucion": "",
"idAlternativaSolucion": 0,
"riesgoIdentificado": "INCUMPLIMIENTO DE LAS CARACTERÍSTICAS DE LOS COMPONENTES",
"idRiesgoIdentificado": 1,
"imgRiesgo1": "JVBERi0xLjcNCiW1tbW1DQoxIDAgb2JqDQo8PC9UeXB1L0NhdGFsb2cvUGFnZXMgMiAwIFI",
"imgRiesgo2": "/9j/4AAQSkZJRgABAQAAQABAAD/2wCEAAEBAQICAgICAgICAgICAgICAgICAgICAgICAgI",
"imgRiesgo3": "/9j/4AAQSkZJRgABAgAAZABkaAD/7AARRHVja3kAAQAEAAAAPAAA/+4ADkFkb2JlAGTAA",
"fechaTerminoEstimado": "2022-06-01 00:00:00.0",
"latitud": "-1",
"longitud": "-1654213",
"txtIpReg": null,
"fechaInicio": null,
"fechaFin": null,
"nomEstado": "RECHAZADO",
"pageIndex": 0,
"pageSize": 0,
"idAvanceFisicoPartida": 0,
"estadoAvance": "1"
},
```

