PERÚ | Ministerio de Transportes y Comunicaciones | Autoridad de Transporte Urbano para Lima y Callao

DIRECTIVA

Código de Documento Normativo	Versión	Documento de Aprobación	Fecha de Aprobación	Páginas
D-001-2023-ATU/GG-OA-UTI	V01	Resolución de Presidencia Ejecutiva N° 059 -2023-ATU/PE		46

DIRECTIVA QUE REGULA LA ATENCIÓN E IMPLEMENTACIÓN DE SISTEMAS INFORMÁTICOS EN LA AUTORIDAD DE TRANSPORTE URBANO PARA LIMA Y CALLAO – ATU

ÍNDICE

DIRECTIVA QUE REGULA LA ATENCIÓN E IMPLEMENTACIÓN DE SISTEMAS INFORMÁTICOS EN LA AUTORIDAD DE TRANSPORTE URBANO PARA LIMA Y CALLAO – ATU

1. OBJETIVO

Establecer disposiciones, criterios y pautas metodológicas para la atención e implementación de sistemas informáticos para la operación y gestión del Sistema Integrado de Transporte de Lima y Callao – SIT, así como para el funcionamiento de la Autoridad de Transporte Urbano para Lima y Callao - ATU.

2. FINALIDAD

Lograr una adecuada implementación de los sistemas informáticos requeridos para mejorar los procesos, contribuir a la toma de decisiones y al logro de los objetivos y metas de la ATU.

3. ALCANCE

Las disposiciones establecidas en la presente Directiva son de cumplimiento obligatorio para las unidades de organización y las/los servidoras/es civiles de la ATU, indistintamente de su nivel jerárquico, régimen laboral o modalidad contractual, que presenten requerimientos de sistemas informáticos relacionados a la gestión y operación de los procesos del SIT; así como, relacionados a los procesos de gestión interna y a los procedimientos administrativos de la ATU, según corresponda.

4. BASE LEGAL

4.1 Ley N° 30900, Ley que crea la Autoridad de Transporte Urbano de Lima y Callao – ATU.

4.2 Ley N° 27658, Ley Marco de Modernización de la Gestión del Estado.

4.3 Decreto Legislativo N° 1412, Decreto Legislativo que aprueba la Ley de Gobierno Digital.

4.4 Texto Único Ordenado de la Ley N° 27444, Ley del Procedimiento Administrativo General, aprobado por Decreto Supremo N° 004-2019-JUS.

4.5 Reglamento de la Ley Marco de Modernización de la Gestión del Estado, aprobado por Decreto Supremo N° 030-2002-PCM.

4.6 Reglamento del Decreto Legislativo N° 1412, Decreto Legislativo que aprueba la Ley de Gobierno Digital, y establece disposiciones sobre las condiciones, requisitos y uso de las tecnologías y medios electrónicos en el procedimiento administrativo, aprobado por Decreto Supremo N° 029-2021-PCM.

4.7 Texto Único Ordenado de la Ley N° 30225, Ley de Contrataciones del Estado, aprobado por Decreto Supremo N° 082-2019-EF.

4.8 Política Nacional de Modernización de la Gestión Pública al 2030, aprobada por Decreto Supremo N° 103-2022-PCM.

4.9 Sección Primera del Reglamento de Organización y Funciones de la Autoridad de Transporte Urbano de Lima y Callao – ATU, aprobada por Decreto Supremo N° 003-2019-MTC.

4.10 Sección Segunda del Reglamento de Organización y Funciones de la Autoridad de Transporte Urbano de Lima y Callao – ATU, aprobada por Resolución Ministerial N° 090-2019-MTC/01.

4.11 Resolución Ministerial N° 004-2016-PCM, que aprueba el uso obligatorio de la Norma Técnica Peruana “NTP ISO/IEC 27001:2014 Tecnología de la Información. Técnicas de Seguridad. Sistemas de Gestión de Seguridad de la Información. Requisitos 2ª Edición”, en todas las entidades integrantes del Sistema Nacional de Transformación digital.

4.12 Resolución Ministerial N° 041-2017-PCM, que aprueba el uso obligatorio de la Norma Técnica Peruana “NTP-ISO/IEC 12207:2016 - Ingeniería de Software y Sistemas. Procesos del ciclo de vida del software. 3ra Edición”.

4.13 Resolución de Presidencia Ejecutiva N° 016-2021-ATU/PE, que aprueba la conformación del Comité de Gobierno Digital de la Autoridad de Transporte Urbano para Lima y Callao – ATU.

La presente normativa incluye sus disposiciones modificatorias, complementarias y prórrogas; así como las normas que las pudieran reemplazar.

5. RESPONSABILIDADES

5.1 Comité de Gobierno Digital de la ATU

- ✓ Aprueba la incorporación y priorización del requerimiento de sistemas informáticos al Plan de Gobierno Digital de la ATU.

5.2 Equipo Técnico de Desarrollo

- ✓ Es responsable de participar activamente en las etapas de desarrollo del sistema de acuerdo al flujo operacional, actividades requeridas y rol establecido, garantizando el cumplimiento de las etapas y actividades dentro de los plazos del cronograma aprobado.

5.3 Oficina de Procesos y Gestión de Riesgos

- ✓ Asiste a la unidad de organización (área usuaria) en el análisis, diagnóstico y modelado de procesos actual y optimizado de acuerdo a la Directiva para la implementación de la gestión por procesos en la ATU vigente.

5.4 Subdirección de Integración y Gestión Tecnológica

- ✓ Es responsable de la atención de los requerimientos de sistemas informáticos relacionados a la gestión y operación de los procesos del SIT y de la supervisión de su implementación.

5.5 Unidad de organización (área usuaria)

- ✓ Es responsable de realizar el requerimiento del sistema informático, solicitar su incorporación al Plan de Gobierno Digital de la ATU en caso se requiera; asimismo, participa activamente en las etapas de implementación, emite conformidad funcional luego de la ejecución de pruebas y da conformidad para el pase a producción.

5.6 Unidad de Tecnologías de la Información

- ✓ Es responsable de la atención de los requerimientos de sistemas informáticos relacionados a los procesos de gestión interna, así como, de los procedimientos administrativos de la ATU y de la supervisión de su implementación.

6. SIGLAS Y GLOSARIO DE TÉRMINOS

Para efectos de la presente Directiva, se utilizan las siguientes siglas y términos:

6.1 SIGLAS:

- **ATU:** Autoridad de Transporte Urbano para Lima y Callao
- **ETD:** Equipo Técnico de Desarrollo
- **OA:** Oficina de Administración
- **OPGR:** Oficina de Procesos y Gestión de Riesgos
- **PGD:** Plan de Gobierno Digital
- **SIGT:** Subdirección de Integración y Gestión Tecnológica
- **SIT**: Sistema Integrado de Transporte de Lima y Callao.
- **TI:** Tecnologías de la Información**.**
- **UTI:** Unidad de Tecnologías de la Información

6.2 GLOSARIO:

- **Atención de requerimiento de sistemas informáticos:** Es el conjunto de actividades que se realizan a lo largo del ciclo de vida del software para

garantizar que los requerimientos sean atendidos de manera efectiva. Estas actividades incluyen: i) Diagnóstico y evaluación; ii) Elección de la modalidad de atención; iii) Análisis de requisitos; iv) Diseño del sistema; v) Construcción - Integración del sistema; vi) Pruebas – Aseguramiento del sistema; vii) Instalación - Operación del sistema; y viii) Mantenimiento del sistema.

- **Cartera de Requerimientos de Sistemas (UTI / SIGT):** Enumeración de requerimiento de sistemas solicitados por las unidades de organización (área usuaria) pendientes de atender por parte de la UTI / SIGT.
- **ETD:** Conformado por el/la representante del área usuaria, el/la responsable/encargado/a del desarrollo y el/la representante de la unidad de organización encargada de la supervisión (UTI / SIGT) para llevar adelante el desarrollo del sistema.
- **Modelamiento del Proceso:** Representación gráfica que permite describir visualmente las actividades implicadas en un proceso, mostrando la relación secuencial entre ellas. La representación de la situación actual se denomina Proceso Actual, mientras que al flujo del proceso mejorado se le denomina Proceso Optimizado.
- **Mesa de Servicio:** Conjunto de recursos humanos y tecnológicos dispuestos por la UTI para atender las solicitudes de incidencias y requerimientos de TI, reportado por las unidades de organización (áreas usuarias).
- **Módulo:** Parte o componente funcional de un sistema informático, que opera individualmente o conectado con el resto de sus componentes.
- **Plan de Adquisiciones:** Enumeración de actividades, plazos, instrumentos y responsables necesarios para la contratación de un servicio de implementación/mantenimiento de un software informático.
- **Plan de Gobierno Digital:** Es el documento elaborado por las entidades de la Administración Pública, el cual define objetivos y proyectos de Gobierno Digital en función de las necesidades de los ciudadanos, necesidades de información y cambios en el entorno, enfocados en la digitalización de servicios públicos, procesos e información de una entidad, haciendo uso intensivo de las tecnologías digitales y la innovación dirigida por datos.
- **Plan de Sistemas de la UTI / SIGT:** Enumeración priorizada elaborada por la UTI / SIGT para la atención de requerimientos de sistemas informáticos solicitados por las unidades de organización (área usuaria), para su ejecución en un periodo de tiempo, no están incorporados en el PGD de la ATU.
- **Personal designado por la unidad de organización:** Personal de la ATU a quien el responsable de la unidad de organización delega la participación en el desarrollo del sistema informático.
- **Prototipo de Interfaces:** Guía visual o prototipo que representa el diseño de las pantallas del sistema informático a desarrollar, su principal objetivo reside en mostrar la facilidad de uso, comportamiento, experiencia de usuario/a y jerarquía de contenidos.
- **Pruebas funcionales del Sistema:** Conjunto de acciones para verificar el comportamiento del sistema informático y comprobar si satisface los requisitos funcionales establecidos por la unidad de organización (área usuaria).
- **Requerimiento de Sistema Informático:** Solicitud sustentada por la unidad de organización (área usuaria) para: i) Implementación de un nuevo sistema o ii) Mantenimiento de un sistema: incorporación de funcionalidades, mejora o correcciones de uno existente.
- **Responsable de la unidad de organización (área usuaria):** Es la máxima autoridad de la unidad de organización, encargado/a de dar a conocer el flujo y secuencia de actividades del proceso, brindar información relevante del mismo, participar en las distintas etapas del desarrollo de un sistema informático y aprobar los entregables funcionales.– Está facultado/a a delegar las citadas responsabilidades a un/una representante de su unidad de organización, de preferencia que sea dueño/dueña del proceso o tenga conocimiento de las actividades del proceso.
- **Responsable/Encargado/a del desarrollo:** Persona natural o jurídica encargada de realizar el servicio de desarrollo de un sistema informático para la ATU.

- **Responsable/Encargado/a del análisis del sistema:** Persona natural o jurídica encargada de realizar la etapa de análisis de requisitos de un sistema informático para la ATU.
- **Responsable/Encargado/a del diseño del sistema:** Persona natural o jurídica encargada de realizar la etapa de diseño de un sistema informático para la ATU.
- **Responsable/Encargado/a de las pruebas del sistema:** Persona natural o jurídica encargada de realizar la etapa de pruebas - aseguramiento de un sistema informático para la ATU.
- **Responsable/Encargado/a de instalación de sistema:** Persona natural o jurídica encargada de realizar la etapa de instalación – operación de un sistema informático para la ATU.
- **Responsable de infraestructura:** Persona natural o jurídica encargada de brindar los componentes de infraestructura tecnológica o ejecutar actividades administrativas y operativas de infraestructura tecnológica necesaria para la operatividad de un sistema informático para la ATU.
- **Responsable de soporte:** Persona natural o jurídica encargada de brindar el soporte a un sistema informático para la ATU.
- **Sistema Informático:** Conjunto de componentes interrelacionados que permite a los/las usuarios/as el almacenamiento, procesamiento y consulta de la información. Sus partes son: hardware, software y las personas que lo usan.
- **Software:** Es el conjunto de componentes lógicos que hace posible la realización de tareas específicas, el mismo que interactuando o dando instrucciones a un componente de hardware hace posible su funcionamiento.
- **Software Público Peruano:** Software o programa de ordenador de titularidad de una entidad de la Administración Pública, cuyo desarrollo es contratado o efectuado directamente por el personal de dicha entidad para soportar sus procesos o servicios, es financiado con fondos públicos, y puede ser puesto a disposición para ser usado, copiado, modificado y distribuido bajo una licencia libre o abierta.
- **Términos de referencia:** Descripción de las características técnicas y las condiciones en que se ejecuta la contratación de servicios en general, consultoría en general y consultoría de obra. En el caso de consultoría, la descripción además incluye los objetivos, las metas o resultados y la extensión del trabajo que se encomienda (actividades), así como si la Entidad debe suministrar información básica, con el objeto de facilitar a los/las proveedores/as de consultoría la preparación de sus ofertas.
- **Unidad de organización (área usuaria):** Órgano o unidad de organización de la ATU que solicita el requerimiento de un sistema informático, representada por el/la responsable o quien este designe.
- **Usuario / Usuaria Interno/a:** Personal de la ATU que hace uso de las funcionalidades del sistema informático, de acuerdo al nivel de acceso autorizado.
- **Usuario / Usuaria Externo/a:** Persona natural o jurídica que hace uso de un sistema informático dispuesto por la ATU para la ciudadanía o administrados/as.
- **Desarrollo o Sistematización de los procesos del SIT:** Es la implementación de sistemas o soluciones tecnológicas a cargo o supervisadas por la SIGT, para soportar los procesos de la operación y gestión del SIT.
- **Infraestructura tecnológica en el marco del SIT:** Infraestructura tecnológica requerida por los procesos del servicio del SIT, cuya solicitud, dimensionamiento e implementación corresponde a la SIGT, delegando su operación y gestión.
- **Desarrollo o Sistematización de los procesos de la ATU:** Es la implementación de sistemas o soluciones tecnológicas desarrolladas internamente o mediante tercerización, supervisadas por la UTI.

7. DISPOSICIONES GENERALES

7.1 De los requerimientos de sistemas informáticos

a) La atención de requerimientos de nuevos sistemas informáticos obedece a la programación establecida en el PGD de la ATU y el Plan de Sistemas de la UTI /

SIGT, la programación puede ser modificada para atender algún requerimiento exigido normativamente o por situaciones excepcionales, debidamente justificadas.

b) La UTI o la SIGT, según corresponda, evalúa la viabilidad e impacto del requerimiento de un nuevo sistema informático antes que sea propuesto para su incorporación en el PGD de la ATU o en el Plan de Sistemas de la UTI / SIGT, velando que esté orientado a la integración y comunicación de los mismos, con la finalidad de no duplicar información y minimizar los problemas de inconsistencia y confiabilidad, confirmando la disponibilidad o solicitando las capacidades de infraestructura tecnológica que necesite.

c) La UTI o la SIGT, según corresponda, evalúa la viabilidad e impacto de los requerimientos referidos a mantenimiento de sistemas, mejoras en las funcionalidades, entre otros, para su incorporación al Plan de Sistemas de la UTI / SIGT.

d) El presupuesto asignado para la atención del requerimiento de un sistema informático considera los recursos para su desarrollo e implementación, así como también la infraestructura tecnológica necesaria para su despliegue, puesta en producción y sostenibilidad durante su ciclo de vida.

e) Los requerimientos cuentan con la evaluación previa de la OPGR para determinar su impacto en el flujo de los procesos.

7.2 De las modalidades de atención del requerimiento.

La atención de un requerimiento de sistema informático se realiza mediante las modalidades siguientes:

(i) Desarrollo propio del sistema informático a cargo de la UTI / SIGT.
(ii) Desarrollo del sistema informático mediante tercerización, bajo la supervisión de la UTI / SIGT.
(iii) Implementación de sistema informático administrativo de propiedad de una entidad o publicado como Software Público Peruano bajo la supervisión o asistencia de la UTI.

7.3 De las etapas, entregables y roles de implementación de sistemas informáticos con desarrollo propio o con tercerización.

a) Las etapas del desarrollo de un sistema informático son las siguientes: i) Diagnóstico y evaluación, ii) Elección de la modalidad de atención; iii) Análisis de Requisitos; iv) Diseño del sistema; v) Construcción - Integración del sistema; vi) Pruebas - Aseguramiento del sistema; vii) Instalación - Operación de sistema; y, viii) Mantenimiento del sistema. A continuación, se detallan el propósito de cada etapa y los entregables que les corresponden:

Cuadro N° 1 Matriz de Etapas, Propósito y Entregables del desarrollo de un sistema informático

N°	ETAPA	PROPÓSITO	ENTREGABLES[1]
1.	Diagnóstico y evaluación	Diagnóstico y evaluación de requerimiento funcional.	✓ Registro en el PGD de la ATU. ✓ Registro en el Plan de sistemas de la UTI / SIGT.
2.	Elección de la modalidad de atención	Definir la modalidad de atención del requerimiento.	✓ Plan de Trabajo y Cronograma UTI. ✓ Plan de Trabajo y Cronograma SIGT. ✓ Plan de Adquisiciones.
3.	Análisis de Requisitos	Establecer los requisitos de los elementos del sistema.	✓ Análisis de requerimiento. ✓ Diagrama de Casos de uso. ✓ Especificaciones de casos de uso. ✓ Plan de trabajo detallado. ✓ Modelado de procesos actual, cuando corresponda.
4.	Diseño del sistema	Elaborar el diseño arquitectural y detallado del sistema para atender los requisitos.	✓ Modelado de procesos optimizado, cuando corresponda. ✓ Requerimiento de infraestructura de TI. ✓ Diagrama de Arquitectura. ✓ Diseño de base de datos. ✓ Prototipo de interfaces de usuario. ✓ Casos de pruebas de usuarios.
5.	Construcción – Integración del sistema	Producir e integrar componentes del sistema alineados al diseño, para satisfacer los requisitos funcionales y no funcionales.	✓ Actas de pruebas funcionales. ✓ Software producido. ✓ Programa fuente del sistema. ✓ Manual de instalación y operación del sistema. ✓ Manual de usuario. ✓ Tratamiento de información histórica cuando corresponda.
6.	Pruebas – Aseguramiento del sistema	Garantizar y verificar el cumplimiento de los requisitos del sistema a nivel funcional y del proceso de desarrollo.	✓ Informe de pruebas del sistema. ✓ Ficha de pase a producción. ✓ Actas de capacitación cuando corresponda.

[1] Los formatos/plantillas de los entregables o equivalente son validados por la UTI se encuentran a disposición en el siguiente enlace: https://atugobpe-my.sharepoint.com/:f:/g/personal/mesadeservicio_atu_gob_pe/EnB39vAUdF9GuVV327cviGgBzN7pNAG78R5YAC_6u52UyQ?e=l6RDP4

N°	ETAPA	PROPÓSITO	ENTREGABLES[1]
7.	Instalación – Operación del sistema	Instalar y operar el sistema, cumpliendo los requisitos en su entorno de producción.	✓ Respaldo del sistema, cuando corresponda. ✓ Instalación del sistema en el entorno de producción. ✓ Solicitud de validación / conformidad de la instalación. ✓ Tratamiento de información histórica, cuando corresponda.
8.	Mantenimiento del sistema	Proveer el soporte para el manejo del cambio, así como efectuar el retiro del sistema al cumplir su ciclo de vida.	✓ Registro de incidentes y soporte del sistema ✓ Informe de retiro del sistema, cuando corresponda.

b) Para la implementación del sistema informático mediante desarrollo propio, la ejecución de las actividades de las etapas del desarrollo están a cargo de la UTI / SIGT; mientras que la supervisión la efectúa el ETD.

c) Para implementación del sistema informático mediante tercerización, el/la proveedor/a se encarga de la ejecución de las etapas del desarrollo del sistema informático bajo la supervisión de la UTI / SIGT. Asimismo, la UTI / SIGT puede intervenir en la ejecución de actividades críticas de aquellas etapas en las que se requiera.

7.4 Implementación de sistema informático administrativo de propiedad de una entidad o publicado como Software Público Peruano.

Para la implementación de sistema administrativo de propiedad de una entidad o publicado como Software Público Peruano, las coordinaciones para la instalación o personalización y supervisión está a cargo de la UTI. Las etapas y roles se detallan en el siguiente cuadro:

Cuadro N° 2 Matriz de Etapas y Roles – Implementación de sistema externo con asistencia de la UTI

N°	ETAPA	PROPOSITO	ROL DESARROLLO	ROL SUPERVISOR
1	Evaluación Funcional	Evaluar el cumplimiento de la funcionalidad del sistema con el proceso o flujo del área usuaria.	Área Usuaria	Área Usuaria
2	Evaluación Técnica	Evaluar la factibilidad técnica de su implementación del sistema en la entidad.	UTI	UTI
3	Instalación – Personalización del sistema	Instalar y personalizar el sistema.	Responsable / UTI	UTI

N°	ETAPA	PROPOSITO	ROL DESARROLLO	ROL SUPERVISOR
4	Pruebas – Aseguramiento del sistema	Garantizar y verificar el cumplimiento de los requisitos del sistema a nivel funcional y del proceso de desarrollo.	Área Usuaria / UTI	UTI
5	Puesta en operación del sistema	Operar el sistema en su entorno de producción	Responsable / UTI	UTI

8. DISPOSICIONES ESPECÍFICAS

8.1 De la solicitud del requerimiento

8.1.1 El área usuaria, a través del/de la responsable de la unidad de organización o a quien éste/a designe, identifica y define las necesidades de sistemas informáticos y efectúa el requerimiento para su atención, conforme al Anexo N° 1 de la presente Directiva. La solicitud de migración de información histórica deberá señalarse en el documento de Análisis de Requerimiento.

8.1.2 El área usuaria, como resultado de la evaluación realizada por la UTI / SIGT al requerimiento, puede solicitar su incorporación al PGD de la ATU mediante el llenado de la Ficha de Proyecto, conforme al Anexo N° 6; de no corresponder su incorporación al PDG de la ATU, se incorpora al Plan de Sistemas de la UTI / SIGT.

8.2 Del diagnóstico y evaluación del requerimiento

8.2.1 La SIGT evalúa la viabilidad de los requerimientos relacionados a la gestión y operación de los procesos del SIT, así como el impacto en los sistemas con los cuales se relaciona o integra.

8.2.2 La UTI evalúa la viabilidad de los requerimientos relacionados a los procesos de gestión interna y a los procedimientos administrativos de la ATU, así como el impacto en los sistemas con los cuales se relaciona o integra.

8.2.3 La UTI / SIGT deriva los requerimientos a la OPGR para su evaluación en cuanto a su impacto en el flujo del proceso. La OPGR efectúa el diagnóstico del proceso, esto es, identifica el problema, las causas y propone alternativas de solución; y en caso que, como resultado de dicho diagnóstico, determine que hay impacto en el flujo de proceso, elabora el modelado de proceso actual y proceso optimizado, cuyo relevamiento se efectúa conjuntamente con el área usuaria.

8.2.4 La UTI / SIGT evalúa cuáles son los requerimientos de implementación de un sistema nuevo que corresponden ser incorporados en el PGD de la ATU. En esos casos coordinan dicha incorporación con el área usuaria.

8.2.5 El Comité de Gobierno Digital de la ATU aprueba y prioriza la incorporación de proyectos al portafolio de proyectos de su PGD.

8.2.6 Los requerimientos de implementación de un sistema nuevo no incorporado en el PGD, así como los requerimientos de mantenimiento de sistemas existentes, son incorporados al Plan de Sistemas de la UTI / SIGT, según corresponda.

8.3 De la elección de la modalidad de atención.

8.3.1 La UTI / SIGT, como resultado de la evaluación, propone al área usuaria la modalidad de atención del requerimiento:

(i) Ejecutar la implementación o actualización del sistema informático mediante desarrollo propio de la UTI / SIGT, lo que conlleva a la elaboración de un Plan de Trabajo y Cronograma de actividades.

(ii) Ejecutar la implementación o actualización del sistema informático mediante tercerización, lo que conlleva a la elaboración de un Plan de Adquisiciones con su respectivo cronograma de ejecución, bajo la supervisión de la UTI / SIGT.

8.3.2 Los requerimientos pueden ser atendidos mediante la implementación y personalización de un sistema informático administrativo de propiedad de una entidad o publicada como Software Público Peruano, con la supervisión y asistencia de la UTI, para lo cual se elabora un Plan de Trabajo y Cronograma de Implementación / personalización.

8.4 Del Plan de Adquisiciones (Tercerización).

8.4.1 Para la elaboración del Plan de Adquisiciones en caso de implementación o actualización del sistema informático mediante la tercerización, se tiene en cuenta lo siguiente:

(i) Se elaboran los Términos de Referencia detallando los requerimientos o requisitos, la unidad de organización (área usuaria) detalla las especificaciones funcionales del sistema informático. La UTI / SIGT, según corresponda, detalla los requerimientos no funcionales: etapas, actividades y entregables contenidos en la presente Directiva, así como metodología, estándares de integración o desarrollo, lineamientos de seguridad de la información y actividades de coordinación.

(ii) En los Términos de Referencia adicionalmente se deben especificar los detalles referidos a derechos de autor de los sistemas producidos, entregables requeridos, plazo de garantía del servicio, soporte de incidentes, así como la coordinación y ejecución de actividades entre el personal responsable del desarrollo y especialistas de la UTI / SIGT.

(iii) La unidad de organización (área usuaria), con la asistencia técnica de la UTI / SIGT, envía a la unidad encargada de la contratación del servicio de tercerización los Términos de Referencia para la selección del/de la proveedor/a responsable del desarrollo tercerizado.

(iv) Adjudicado/a o notificado/a al/a la responsable del desarrollo tercerizado, la UTI / SIGT en coordinación con la unidad de organización (área usuaria), solicita a la unidad encargada de la contratación la documentación vinculada al proceso de selección: Bases, Propuesta y Contrato para su ejecución y supervisión del servicio.

(v) En base a los Términos de Referencia, el/la responsable/encargado/a del desarrollo del sistema informático identifica las actividades y presenta un cronograma para la ejecución del servicio. Se realiza la reunión inicial del servicio con la participación del ETD, en dicha reunión el/la responsable encargado/a del desarrollo del sistema informativo entrega el cronograma de actividades y el/la responsable de la unidad de organización (área usuaria) valida y brinda conformidad al cronograma.

(vi) En el cronograma de actividades, que detalla los plazos, actividades, entregables y responsables, se ejecutan según lo programado. La unidad de organización (área usuaria) y la UTI / SIGT establecen reuniones periódicas para su monitoreo y seguimiento.

(vii) En base a los Términos de Referencia y al cronograma de actividades para la ejecución del servicio, previa opinión técnica favorable de la UTI / SIGT, la unidad de organización (área usuaria) emite la(s) conformidad(es) correspondiente(s) a los entregables presentados por el/la responsable encargado/a del desarrollo del sistema informático.

8.5 Del Análisis de requisitos.

8.5.1 Para establecer los requisitos de los elementos del sistema, se efectúa lo siguiente:

(i) En los casos de requerimientos con impacto en el proceso, se debe contar con el modelado del proceso actual elaborado por la OPGR.

(ii) El/la responsable de análisis de la UTI / SIGT o encargado/a del análisis del sistema elabora los documentos:

- ✓ Análisis de Requerimiento del sistema, conteniendo:
  - Requerimientos funcionales.
  - Requerimientos no funcionales.
  - Requerimientos de Infraestructura.
  - Requerimientos de Seguridad.
  - Requerimiento de protección de datos.
- ✓ Diagramas de Caso de Uso del sistema.
- ✓ Especificaciones de Casos de Uso
  - Diagrama de actividades.
- ✓ Plan de trabajo detallado.

(iii) Los integrantes del ETD efectúan la revisión del Plan de Trabajo detallado y al brindar su conformidad se comprometen a ejecutar las actividades programadas, actividades necesarias y las que deriven para el desarrollo del sistema en el plazo acordado.

8.6 Del Diseño del sistema.

8.6.1 Para elaborar el diseño arquitectural y detallado para atender los requisitos del sistema, se efectúa lo siguiente:

(i) En los casos de requerimientos con impacto en el proceso, se debe contar con el modelado del proceso optimizado elaborado por la OPGR.

(ii) El/la responsable de diseño de la UTI / SIGT o encargado/a del diseño del sistema elabora los documentos:

- ✓ Diagrama de Arquitectura.
- ✓ Diseño de la base de datos.
  - Diagrama de clases.
  - Diagrama entidad relación.
  - Diccionario de datos.
- ✓ Prototipos de interfaces de usuario.
- ✓ Casos de pruebas de usuarios.
- ✓ Requerimiento de infraestructura de TI.

(iii) El personal designado por la unidad de organización (área usuaria) comunica al/a la analista funcional de la UTI / SIGT o al/a la responsable del diseño del sistema su conformidad con los prototipos de interfase de usuario.

8.7 De la construcción – integración del sistema.

8.7.1 Para producir e integrar los componentes del sistema diseñados para satisfacer los requisitos funcionales y no funcionales, se efectúa lo siguiente:

(i) El/la responsable de desarrollo de sistemas de la UTI / SIGT o encargado/a del desarrollo del sistema elabora los documentos o hace entrega de los siguientes productos:
- ✓ Actas de conformidad de pruebas funcionales.
- ✓ Software producido.
- ✓ Programas fuente del sistema.
- ✓ Manuales de usuario y técnicos.

(ii) El/la analista funcional de la UTI / SIGT y el personal designado por la unidad de organización (área usuaria) confirman que las funcionalidades del sistema informático desarrollado están de acuerdo a su requerimiento, mediante la firma del Anexo N° 2 de la presente Directiva.

8.8 De las pruebas – aseguramiento del sistema.

8.8.1 Para garantizar y verificar el cumplimiento de los requisitos del sistema a nivel funcional y del proceso de desarrollo, se efectúa lo siguiente:

(i) Para su ejecución, el/la responsable de las pruebas de la UTI / SIGT o encargado/a de pruebas del sistema debe disponer como mínimo de los siguientes entregables provenientes de las etapas anteriores: Acta de Pruebas Funcionales (Anexo 2), Casos de pruebas de usuarios, Diagrama de la arquitectura, Programa fuente del sistema, Manual de instalación y operación del sistema, y Software producido .

(ii) El/la responsable de pruebas de la UTI / SIGT o encargado/a de efectuar las pruebas del sistema elabora los documentos / productos:
- ✓ Pruebas funcionales del sistema
- ✓ Pruebas de seguridad del sistema
- ✓ Informe de pruebas del sistema
- ✓ Ficha de pase a producción.
- ✓ Actas de capacitación, cuando corresponda.

(iii) El/la responsable de desarrollo de la UTI / SIGT, el personal designado por la unidad de organización (área usuaria) y el/la responsable de las pruebas de calidad del sistema confirman que se han superado las pruebas y aseguramiento del sistema, mediante la firma de la Ficha de pase a producción (Anexo N° 3) de la presente Directiva, y solicitan al/a la responsable de infraestructura de la UTI / SIGT o encargado/a de instalación de sistema continue con el proceso.

8.9 De la instalación – operación del sistema.

Para instalar y operar el sistema cumpliendo los requisitos en su entorno de producción, se efectúa lo siguiente:

(i) El/la responsable de instalación de la UTI / SIGT o encargado/a de la instalación de sistema ejecuta las siguientes actividades / productos:
- ✓ Respaldo del sistema, cuando corresponda.
- ✓ Instalación del sistema en el entorno de producción.
- ✓ Solicitud de validación / conformidad de la instalación.

(ii) El/la responsable de desarrollo de la UTI / SIGT o el encargado del desarrollo del sistema verifica la correcta instalación del sistema y brinda conformidad a la actividad de instalación; asimismo, comunica a la unidad de organización (área usuaria) la operatividad y disponibilidad del sistema para su uso.

(iii) Para los requerimientos que contemplan migración de información histórica, la unidad de organización (área usuaria) y la UTI / SIGT evalúan dicha información, planifican y ejecutan actividades para su entrega, carga y verificación.

8.10 Del mantenimiento del sistema.

8.10.1 Para proporcionar el soporte para el manejo del cambio, y en el caso se requiera, efectuar el retiro del sistema al cumplir su ciclo de vida, se efectúa lo siguiente:

(i) El/la responsable de soporte de la UTI / SIGT ejecuta el:
- ✓ Registro de incidentes y soporte del sistema.

(ii) El/la responsable de desarrollo de sistema de la UTI / SIGT elabora el:
- ✓ Informe de retiro del sistema, en el caso que corresponda.

(iii) El soporte del sistema debe eliminar los defectos detectados durante la operatividad del sistema, evaluar las condiciones adecuadas de la infraestructura, identificar y gestionar los riesgos operacionales para el funcionamiento del sistema. Asimismo, se debe proporcionar la asistencia técnica y de apoyo a las unidades de organización en relación con el uso del sistema informático.

(iv) La unidad de organización (área usuaria) es responsable de comunicar las incidencias a la UTI / SIGT para su atención.

(v) Las mejoras a un sistema recientemente entregado se solicitan a la UTI / SIGT como un nuevo requerimiento, utilizando el Anexo N° 1 de la presente Directiva.

8.11 Del seguimiento y monitoreo de la implementación del sistema informático.

8.11.1 El seguimiento y monitoreo de la implementación del sistema informático se efectúan mediante reuniones semanales convocadas por la UTI / SIGT, en las que interviene el ETD y otros responsables o unidades de organización que estés involucrados o tengan interés en la implementación, a fin de verificar el cumplimiento del cronograma y brindar apoyo en eliminar impedimentos que interrumpen el desarrollo.

8.11.2 Las observaciones, subsanaciones, validaciones y conformidades realizadas por los/las responsables sobre la documentación producida en el desarrollo del sistema, deben efectuarse dentro del plazo especificado para cada etapa/actividad en el cronograma de actividades.

8.12 De la Capacitación y Documentación

8.12.1 La programación y ejecución de las capacitaciones a cargo del/de la responsable/encargado/a del desarrollo del sistema informático debe formar parte del cronograma de actividades.

8.12.2 La UTI / SIGT o el/la responsable/encargado/a del desarrollo del sistema informático, solicita a al/a la responsable de la unidad de organización (área usuaria) la lista de asistentes convocados/as para la capacitación.

8.12.3 Las capacitaciones virtuales o presenciales para explicar el uso del sistema, brindadas por la unidad de organización (área usuaria) a usuarios externos de la ATU se realizan en coordinación con la UTI / SIGT.

8.12.4 El/la responsable/encargado/a del desarrollo del sistema informático provee los Manuales del Sistema Informático son provistos, en medio físico y digital editable, al Área Usuaria y a la UTI / SIGT.

8.13 De la Seguridad y Privacidad

8.13.1 Los accesos que se habilitan a los/las usuarios/as para los sistemas desarrollados cumplen con la Política Institucional de Seguridad de la Información de la ATU.

8.13.2 La unidad de organización (área usuaria) y la UTI / SIGT verifican que el/la responsable o encargado/a del desarrollo de sistema cumpla los lineamientos anexos a la presente Directiva, para salvaguardar su carácter reservado y confidencial, además de evitar la difusión a personas no autorizadas de la

documentación institucional con la que interactúa durante todo el proceso de desarrollo del sistema informático, así como con los acuerdos de confidencialidad y derechos de autor especificados en los Términos de Referencia del servicio y la Política Institucional de Seguridad de la Información de la ATU.

8.14 Del uso de estándares para el desarrollo de sistemas informáticos

Para la implementación de sistemas informáticos con desarrollo propio a cargo de la UTI / SIGT, así como en los desarrollos mediante tercerización bajo la supervisión de la UTI / SIGT, los/las responsables/encargados/as del desarrollo ejecutan las actividades teniendo en cuenta los siguientes lineamientos y guías:

a) Lineamientos para el desarrollo de sistemas (Anexo 4 de la presente Directiva).
b) Lineamientos de seguridad para el desarrollo de sistemas (Anexo 5 de la presente Directiva).
c) Guía de Diseño y Programación de Base de Datos - Oracle y SQL Server[2].

La revisión de los Anexos de la presente Directiva se efectúa de manera coordinada entre la UTI / SIGT, debiendo proponer su actualización en caso corresponda de acuerdo a la normativa de la materia.

9. ANEXOS

9.1 Anexo 1: Formato: “Solicitud de Requerimiento de Sistema Informático”.
9.2 Anexo 2: Formato: “Acta de Pruebas funcionales”.
9.3 Anexo 3: Formato: “Ficha de Pase a Producción”.
9.4 Anexo 4: Lineamientos para el desarrollo de sistemas.
9.5 Anexo 5: Lineamientos de seguridad para el desarrollo de sistemas.
9.6 Anexo 6: Portafolio del Plan de Gobierno Digital – Ficha de Proyecto
9.7 Anexo 7: Diagrama de Flujo.

[2] Enlace: https://atugobpe-my.sharepoint.com/:b:/g/personal/mesadeservicio_atu_gob_pe/ET7eyvD-tw9KgYM_SOcnDfkBGTgXo_ca1IyZTLqbl0oVpg?e=vlvg0A

ANEXO 1:

SOLICITUD DE REQUERIMIENTO DE SISTEMA INFORMÁTICO

1. INFORMACION GENERAL

Área usuaria:		Fecha y Hora:	
Responsable:			
Sistema:			
Módulo/Fase:			

2. TIPO DE REQUERIMIENTO: Marque con un aspa "X" según corresponda:

☐ Nuevo Sistema / aplicativo	☐ Nueva funcionalidad	☐ Otro ( )

3. DESCRIPCIÓN DEL REQUERIMIENTO

4. ORIGEN - JUSTIFICACIÓN

Listado de proyectos PGD		Actividad del POI		Priorización Alta Dirección	

5. CRITICIDAD: Marque con un aspa "X" según corresponda:

☐ Alta	☐ Media	☐ Baja

6. APROBACIÓN

(Firma) ---------- Nombres y Apellidos del solicitante	(Firma) ---------- *Nombres y Apellidos del jefe de Área*

	Versión: *1.0*
Preparado por: OA - UTI	***Página:*** *1 de 1*

ANEXO 2:

ACTA DE PRUEBAS FUNCIONALES

<NOMBRE DEL SISTEMA INFORMÁTICO>

Fecha: __/__/____

1. INFORMACIÓN GENERAL

SISTEMA	
MÓDULO	
ENLACE O REFERENCIA	
ANALISTA	
ÁREA USUARIA	
USUARIO FUNCIONAL	

2. RESULTADOS DE LA PRUEBA

ID REQUERIMIENTO	DESCRIPCIÓN	CONFORME (SI/NO)

3. COMENTARIOS

ID REQUERIMIENTO	DESCRIPCIÓN

4. APROBACIÓN

Nombres y Apellidos del Analista Funcional	Nombres y Apellidos del jefe de Área

	Versión: *1.0*
Preparado por: OA - UTI	***Página:*** *1 de 1*

ANEXO 3:

FICHA DE PASE A PRODUCCIÓN

<NOMBRE DEL SISTEMA INFORMÁTICO>

Fecha: __/__/____

1. INFORMACIÓN GENERAL

SISTEMA	
MÓDULO	
ACCIÓN	
ANALISTA	

2. DETALLES DEL PASE

Servidor	
Directorio	
Nombres de Ejecutable	
Ruta Fuente	
Ruta Destino	
Otros archivos	
Observaciones	

3. RESPONSABLES

Analista Funcional	Responsable de Pruebas QA	Responsable Funcional

Impactó la seguridad de información	**Si**		**No**	
De ser el caso afirmativo, indicar los controles implementados en el sistema				

4. APROBACIÓN

Nombres y Apellidos Responsable	Nombres y Apellidos del jefe de Área Usuaria	Nombres y Apellidos Responsable pruebas de calidad

de Desarrollo		

ANEXO 4:

LINEAMIENTOS PARA EL DESARROLLO DE SISTEMAS

ÍNDICE

LINEAMIENTOS PARA EL DESARROLLO DE SISTEMAS

Los lineamientos para el desarrollo de sistemas informáticos y aplicaciones con la finalidad de uniformizar los criterios y las tecnologías empleadas para su estandarización en la ATU son las siguientes:

1. Principios para el desarrollo de software

El software se desarrolla bajo los principios SOLID:

1.1. Principio de responsabilidad única

Cada clase o módulo debe tener solo una responsabilidad dentro de la aplicación (alta cohesión). Si se identifica más de un propósito en una de las clases, se debe considerar separar dicha funcionalidad en otra clase. Nótese que el principio no propone tener un solo método en cada clase, sino que todos sus métodos sirvan a un solo propósito.

1.2. Principio de abierto/cerrado

Principio que permite heredar el comportamiento de una entidad de software sin modificarlo. Es decir, los módulos deben estar abiertos para su extensión, pero cerrados para su modificación. Este principio es muy importante al definir las clases, paquetes o módulos.

1.3. Principio de substitución de Liskov

Los objetos deben poder ser remplazados con instancias de sus subtipos sin alterar la definición del sistema. En otras palabras, los objetos derivados de una clase deben poder ser remplazados por un objeto de la clase base, sin modificar el comportamiento de la misma. La solución es diseñar por contrato (o interfaz).

1.4. Principio de segregación de la interfaz

Las interfaces se diseñan con propósitos específicos o con un solo rol en lugar de diseñar pocas interfaces generales que obligue a implementar métodos que no se necesita. De este modo favorecemos la creación de software con alta cohesión y bajo acoplamiento.

1.5. Principio de inversión de dependencia

El código debe depender de abstracciones. Los módulos de alto nivel no deben depender de los módulos de bajo nivel; ambos deben depender de abstracciones. Las abstracciones no deben depender de “detalles” o de clases particulares.

2. Buenas prácticas

2.1 Nombre de las clases

- El nombre de las clases debe ser redactado en singular y guardar concordancia con su propósito.
- Evitar nombres excesivamente largos.

2.2 Nombre de los métodos

- El nombre de las funciones debe comenzar con un verbo en infinitivo, seguido del motivo de la función. Ejemplo: “generarReportePagos”.
- El nombre debe tener sentido con la finalidad del método, siendo compresible y auto explicativo.
- Evitar nombres excesivamente largos.

2.3 Codificar clases pequeñas

- Una clase debe manejar una entidad conceptual y tener definida la funcionalidad por la cual fue creada.
- Se debe enfocar en realizar pocas funciones de manera efectiva.

2.4 Codificar métodos pequeños

- Los métodos se deben enfocar en realizar una única tarea o actividad.
- Se debe manejar control de errores y enviar respuestas claras sobre lo ocurrido.

2.5 Usar librerías estándar

Es recomendable usar librerías estándar que ya han sido aprobadas, depuradas y usadas en los desarrollos, comprobando las versiones y vulnerabilidades de las dependencias antes de utilizarlas.

2.6 Documentar código

Se debe documentar los métodos y clases utilizando herramientas estándar, y debe ser actualizada según cambie el código.

2.7 Comentar el código

Se debe evitar comentarios que no aportan valor, manteniendo el código limpio.

2.8 Utilizar sistema de codificación

Los archivos del proyecto deben utilizar la codificación UTF-8.

3. Arquitectura

La estructura de los proyectos de desarrollo de software debe seguir el patrón de arquitectura Domain Driver Design (DDD) que se refiere a la distribución lógica de los conceptos y la forma en como está organizado y estructura el código.

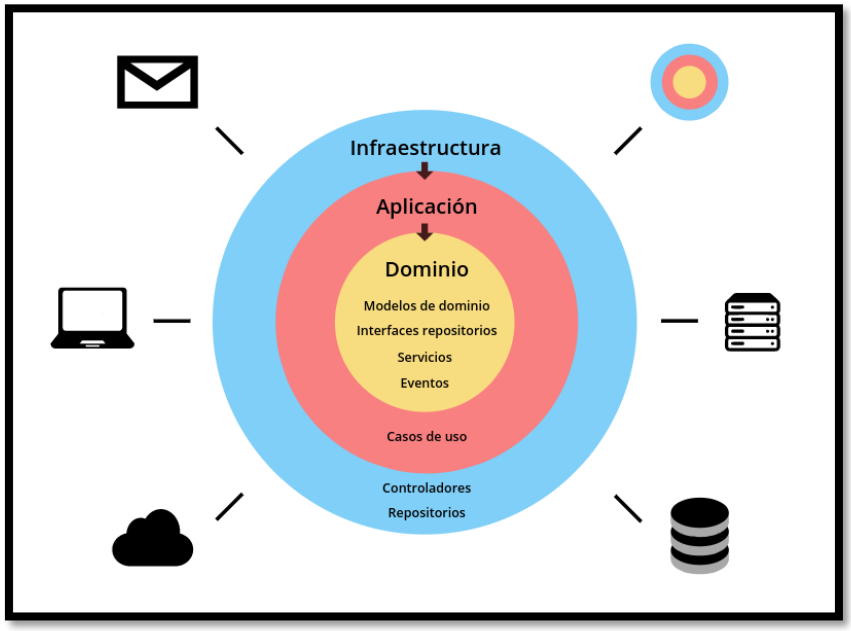


Figura 1. Arquitectura DDD básica para los proyectos de desarrollo de software

Como se aprecia en la Figura 1, en esta arquitectura los conceptos interactúan únicamente con sus capas inferiores manteniendo la responsabilidad y permitiendo abstraer funcionalidades de una clase en concreto.

3.1. Dominio

El dominio contiene las entidades conceptuales, interfaces de repositorios y definición de métodos del negocio.

3.2. Aplicación

La aplicación contiene los casos de uso que albergan la lógica de negocio utilizando las entidades conceptuales definidas y los métodos del negocio.

3.3. Infraestructura

La infraestructura contiene los controladores, implementación de los repositorios, adaptadores para comunicarse con dependencias como la base de datos, correo, etc.

3.4. Vista

La vista contiene los componentes gráficos de la aplicación como HTML y CSS, sirviéndose de las funciones proporcionadas por la infraestructura.

3.5. Cliente

El/la cliente es el/la usuario/a final que interactúa con la aplicación a través de un browser o navegador de internet con las funcionalidades desarrolladas.

4. Estructura de un proyecto

Las Figuras 2, 3 y 4 muestran ejemplos de la estructura de directorio para el desarrollo de proyectos:

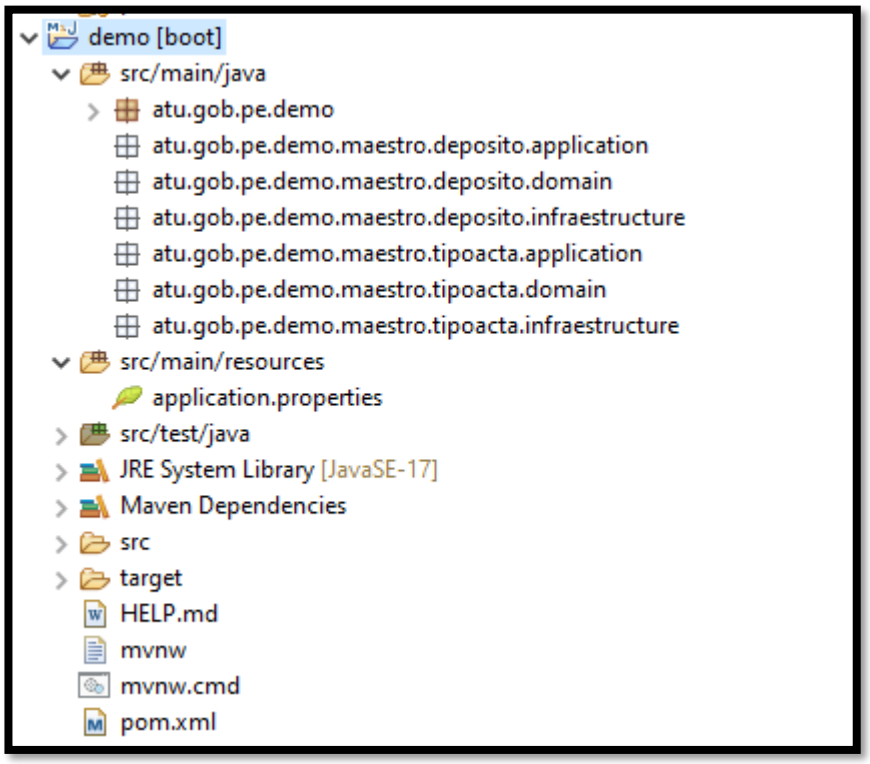


Figura 2. Ejemplo de estructura de un proyecto de software backend en Java.

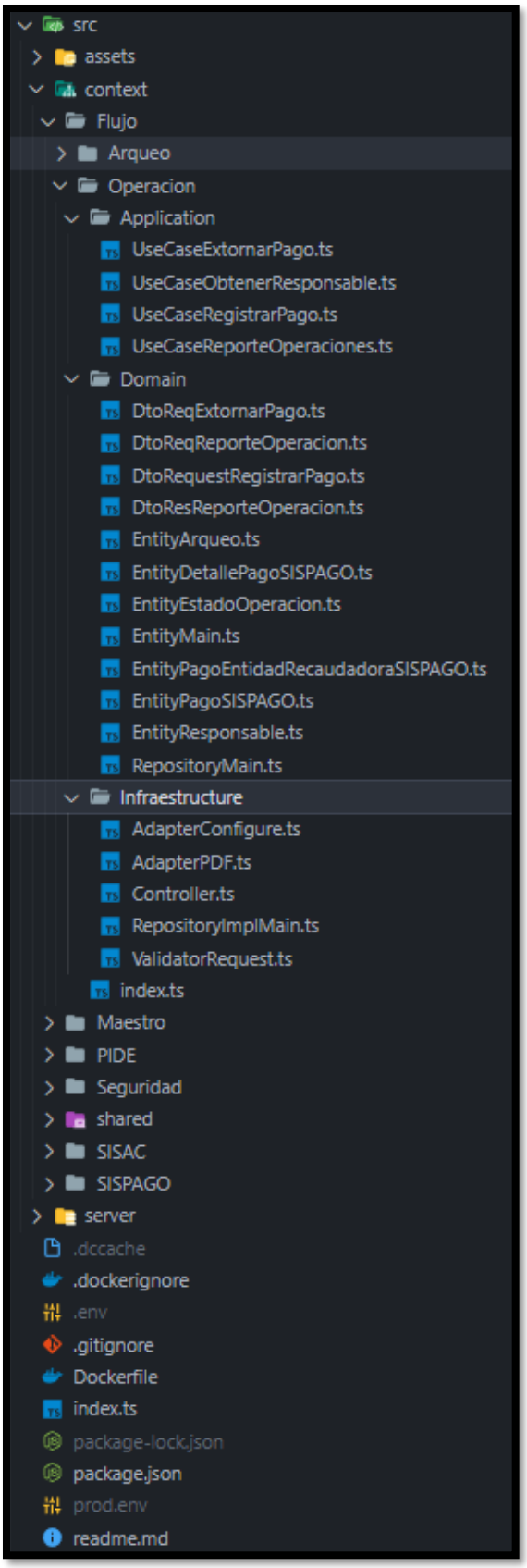


Figura 3. Ejemplo de estructura de un proyecto de software backend en NodeJS.

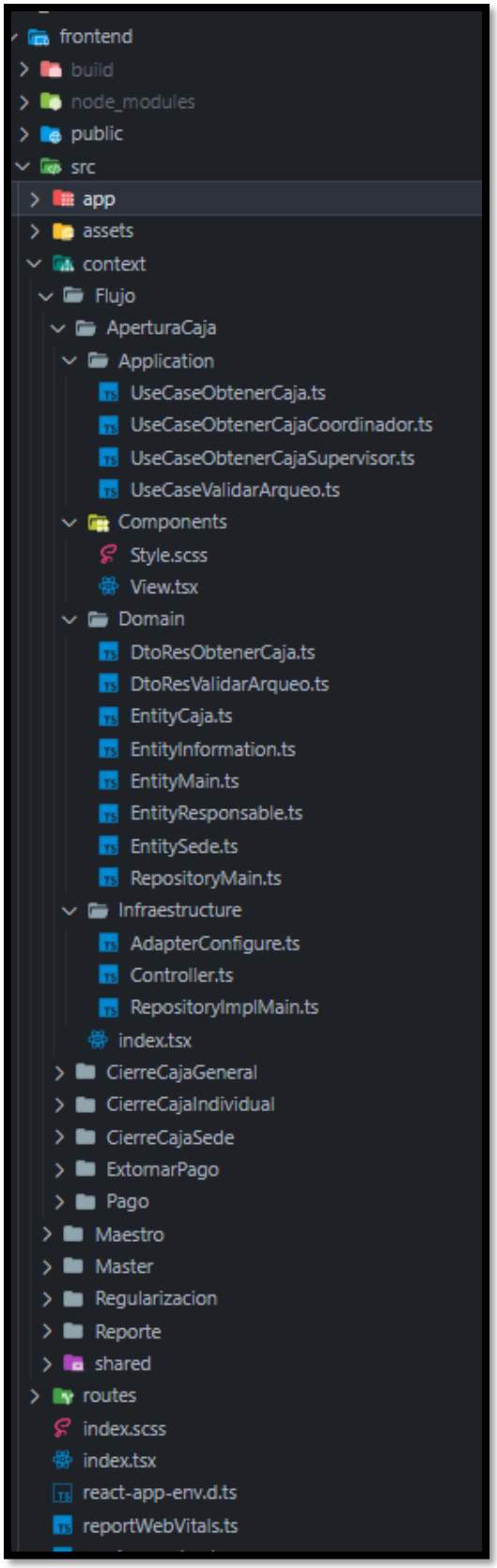


Figura 4. Ejemplo de estructura de un proyecto de software frontend en ReactJS.

La Figura 2, muestra la estructura de directorios del proyecto con todos los módulos en Java.

Las Figuras 3 y 4 muestran la estructura de directorios del proyecto con los módulos y componentes con NodeJS y ReactJS.

Los módulos y componentes se basan en los principios SOLID y la arquitectura DDD indicada en el presente documento en la Figura 1.

5. Tecnologías para el desarrollo de software

Los proyectos de desarrollo de software que se implementen en la Autoridad de Transporte Urbano para Lima y Callao se desarrollan considerando las siguientes tecnologías, según corresponda:

5.1. Gestor de dependencias y construcción de proyectos

Tecnología	Provider	Versión
Maven	Apache	3.x o superior
NPM	NodeJS	8.x o superior

5.2. Lenguajes de programación

Lenguaje	Provider	Versión
Java	Oracle	8.x o 17.x
TypeScript	Microsoft	4.9.x

5.3. Frameworks Backend

Framework	Provider	Versión
Spring Boot	Pivotal	2.x o 3.x
Express JS	OpenJS Foundation	4.x o superior

5.4. Frameworks Frontend

Framework	Provider	Versión
Angular	Google	10.x o superior
React JS	Meta Open Source	18.x o superior
Bootstrap	Bootstrap Team	5.x

5.5. Dependencias para reportes

Tipo	Librería	Versión
PDF	Jasperreports	6.18.x o superior
PDF	PDFMake	0.2.x o superior
Excel	poi.ooxml	5.2.x o superior
Excel	ExcelJS	4.3.x o superior

5.6. Dependencias para logging

Librería	Versión
spring-boot-starter-log4j2	2.x o superior
Morgan	1.10.x o superior

6. Plataforma de despliegue

Infraestructura	Provider	Versión
OPENJDK 11	Oracle	11.x o superor
NodeJS	OpenJS Foundation	18.12.x o superior
Docker	Docker Inc	20.x o superior
Docker Compose	Docker Inc	2.12.x o superior

7. Estándares de desarrollo

Los proyectos de desarrollo de software deberán contar con la siguiente nomenclatura:

atu.gob.pe.{nombre de proyecto}.{versión} Empaquetado Backend: jar, tg, tgz, sourcecode (Según lo requerido)

7.1. Definición de variables

a) Las variables se definen de acuerdo al estándar CamelCase, donde la primera letra debe estar en minúscula.

b) Las variables de tipo constantes se definen completamente en mayúscula, separando cada palabra por un guion bajo para su definición.

Java	TypeScript
private String nombre; private static final String LOG_LEVEL = “tiny”; public static final int SIZE = 4;	private nombre: string; private readonly LOG_LEVEL: string = “tiny”; public readonly SIZE: number = 4;

c) Se recomienda realizar una declaración por línea para mejor el entendimiento.

d) Se debe documentar cada variable local con un comentario al final de la línea con una breve explicación de su uso:

Java	TypeScript

String nombre; // Nombre del Deposito
int size = 4; // Capacidad del Deposito
	let nombre: string; // Nombre del Deposito
	let size: number = 4; // Capacidad del Deposito

e) Para definir una variable de tipo Lista, Arreglo, Colección o elementos derivados que almacenen un conjunto de variables, se define en plural representando los tipos de datos almacenados.

Java	TypeScript
String[] nombres; List medidas;	let nombres: string[]; let medidas: Array<number>;

f) Evitar declarar más de una variable en la misma línea

Java	TypeScript
String nombre, primerApellido;	let nombre: string, primerApellido: string;

g) Se debe inicializar las variables locales donde son declaradas.

Java	TypeScript

void myMethod() {
int variable1 = 0; // Inicio bloque método
if (condition) {
	myMethod(): void {
	let variable1: number = 0; // Inicio bloque método
	if (condition) {

int variable2 = 0; // Inicio
bloque condición
}
}
	let variable2: number = 0; // Inicio
	bloque condición
	}
	}

h) Para la definición de variables de bucles se utiliza las letras i, j, k, etc.; dado que se considera un estándar ampliamente utilizado.

Java	TypeScript
for (int i = 0; i < maxLength; i++) {}	for (let i: number = 0; i < maxLength; i++) {}

i) Se utiliza las siguientes indicaciones para codificar clases e interfaces:

- Evitar espacios entre el nombre del método y paréntesis que inicia los parámetros.
- La llave inicial se coloca al final de la misma línea de la declaración.
- La llave final se coloca en una línea al final.
- Los métodos están separados por una línea en blanco.

Java	TypeScript

class Example extends
Object {
int field1;
int field2;
 	
public Sample(int i, int j) {
field1 = i;
field2 = j;
}
 	
int emptyMethod() {
}
…
}
	class Example extends Object {
	field1: number;
	field2: number;
	
	constructor(i: number, j: number) {
	this.field1 = i;
	this.field2 = j;
	}
	
	public emptyMethod(): number {
	}
	…
	}

j) Para la definición del control de excepciones, es un estándar ampliamente utilizado, el uso de la letra “e” para definir la excepción.

Java	TypeScript

try {
…
} catch (Exception e) {
} finally {}
	try {
	…
	} catch (e) {
	if (e instanceof Error) {}
	} finally {}

k) Cada línea contiene una sola sentencia.

Java	TypeScript

argv++;
argc--;
argv++; argc--; // Incorrecto
	argv++;
	argc--;
	argv++; argc--; // Incorrecto

l) El valor retornado de un método no utiliza paréntesis a menos que sea necesario para un mejor entendimiento.

Java	TypeScript
return; return myClass.calcularIGV(); return (size ? size : defaultSize);	return; return myClass.calcularIGV(); return (size ? size : defaultSize);

m) Las sentencias “if”, “elseif” y “else” tienen la siguiente estructura.

Java	TypeScript

if (condition) {
…
} else if (anotherCondition) {
...
} else {
…
}
	if (condition) {
	…
	} else if (anotherCondition) {
	...
	} else {
	…
	}

n) La sentencia “for” tiene la siguiente estructura.

Java	TypeScript

for (inicialización; condición; actualización) {
…
}
 	
for (Object row: lista) {
…
}
	for (inicialización; condición; actualización) {
	…
	}
	
	for (let row of lista) {
	…
	}

o) La sentencia “switch” tiene la siguiente estructura.

Java	TypeScript

switch (value) {
case ABC:
…
break;
case XYZ:
…
break;
default:
…
break;
}
	switch (value) {
	case ABC:
	…
	break;
	case XYZ:
	…
	break;
	default:
	…
	break;
	}

p) La sentencia “try-catch” siempre debe implementar “finally”, aunque no ejecute código dentro de ella.

Java	TypeScript

try {
…
} catch (Exception e) {
…
} finally {
…
	try {
	…
	} catch (e) {
	…
	} finally {
	…

}	}

7.2. Definición de paquetes

- Los nombres de paquetes tienen que ser definidos en singular y en letras minúsculas.
- La estructura raíz de paquetes es la siguiente: atu.gob.pe.

7.3. Definición de clases

– El nombre de la clase es un sustantivo, cuya primera letra está en mayúscula y sigue el estándar de la nomenclatura “CamelCase”.
– Se puede utilizar mayúsculas en palabras que sean siglas.
– No se debe incluir guiones o guiones bajos entre palabras.
– Utilizar nombres simples y descriptivos
– Evitar abreviaturas o acrónimos.

7.4. Definición de propiedades

– Se declara las propiedades de una clase en el siguiente orden:
  - Públicos (public)
  - Protegidos (protected)
  - Privados (private)

7.5. Definición de los métodos

– Se declara los métodos de una clase en el siguiente orden:
  - Constructores
  - Métodos públicos
  - Métodos protegidos
  - Métodos privados
– Los métodos empiezan con minúscula, indicar el verbo en infinitivo de la operación o proceso que realizar y seguir la nomenclatura “CamelCase”.
– Se documenta los métodos definidos tanto como los parámetros de entrada y el resultado.

8. Estándares de construcción

8.1. Estructura del sistema de archivos

El sistema de archivos de los sistemas está gestionado a través de un servicio de FTP, en donde se debe estructurar de la siguiente forma:
/{**RUTA_FTP**}/{**ALIAS_SISTEMA**}/{**ALIAS_MODULO**}/{**ALIAS_ENTIDAD**}/{**NOMBRE_ARCHIVO**}

Ejemplo:
/ATU/SISCAJA/COBRANZA/OPERACION/Ticket00001.pdf

8.2. Servicios REST

– Los servicios REST que se expongan de manera interna o externa, deben seguir la siguiente nomenclatura:

https://api.atu.gob.pe/api/{**alias_sistema**}/{**alias_modulo**}/{**alias_entidad**}

Por ejemplo:

https://api.atu.gob.pe/api/SISCAJA/Cobranza/Arqueo

– Los servicios deben estar mapeados con el método HTTP que corresponde según la acción que realiza:

Método	URL	Descripción
GET	https://api.atu.gob.pe/SISCAJA/Cobranza/Arqueo	Listar todos los arqueos
POST	https://api.atu.gob.pe/SISCAJA/Cobranza/Arqueo/find	Listar arqueos con parámetros de filtro
POST	https://api.atu.gob.pe/SISCAJA/Cobranza/Arqueo/saveOne	Registrar un arqueo
PATCH	https://api.atu.gob.pe/SISCAJA/Cobranza/Arqueo/updateOne	Actualizar un arqueo
DELETE	https://api.atu.gob.pe/SISCAJA/Cobranza/Arqueo/deleteOne	Eliminar un arqueo

– El tipo de respuesta de los servicios REST debe ser indicado en el Content Type de la cabecera de respuesta.

Ejemplo:

- application/json
- application/pdf
- application/xml

– Se debe usar los siguientes códigos para el HTTP STATUS de la respuesta de un servicio REST:

- 200: Petición ejecutada correctamente.
- 4xx: Petición no ejecutada por errores del cliente.
- 5xx: Petición no ejecutada por errores del servidor.

– En el caso de las peticiones ejecutadas correctamente.

http status: 200

```
{
// body response
}
```

– En el caso de las peticiones no ejecutadas por errores.

```
http status: 400
{
“code”: <código de error>,
“message”: <nombre de error>,
“description”: <descripción de error>,
}

http status: 500
{
“code”: <código de error>,
“message”: <nombre de error>,
“description”: <descripción de error>,
}
```

ANEXO 5:

LINEAMIENTOS DE SEGURIDAD PARA EL DESARROLLO DE SISTEMAS

ÍNDICE

LINEAMIENTOS DE SEGURIDAD PARA EL DESARROLLO DE SISTEMAS

Los lineamientos de seguridad digital que se aplican en los sistemas informáticos y aplicaciones desarrolladas internamente como para las adquiridas a proveedores en la ATU, son las siguientes:

1. Validación de entradas de datos

- ✓ Realizar todas las validaciones de datos en un sistema confiable (por ejemplo: el servidor).
- ✓ Identificar todas las fuentes de datos y clasificarlos como confiables o no confiables. Validar todos los datos provenientes de fuentes no confiables (por ejemplo: bases de datos, secuencias de archivo, etc.).
- ✓ Debería existir una rutina de validación de datos de entrada centralizada para la aplicación.
- ✓ Especificar sets de caracteres apropiados, tales como UTF-8 (formato de codificación de unicode), para todas las fuentes de entrada.
- ✓ Codificar los datos a un set de caracteres común antes de validar (Canonicalización).
- ✓ Todas las fallas en la validación deber terminar en el rechazo del dato de entrada.
- ✓ Determinar si el sistema soportará sets de caracteres UTF-8 extendidos y de ser así, validarlos luego de terminada la decodificación del UTF-8.
- ✓ Validar todos los datos brindados por el cliente antes de procesarlos, incluyendo todos los parámetros, URLs y contenidos de cabeceras HTTP (por ejemplo, nombres de Cookies y valores). Asegurarse de incluir post backs automáticos desde JavaScript, Flash u otro código embebido.
- ✓ Verificar que los valores de la cabecera tanto en solicitudes como en respuestas contengan solo caracteres ASCII.
- ✓ Validar datos redireccionados (un atacante puede enviar contenido malicioso directamente en el destino de la redirección, eludiendo entonces la lógica de la aplicación y cualquier otra validación realizada antes de la redirección).
- ✓ Validar tipos de datos no esperados.
- ✓ Validar rangos de datos.
- ✓ Validar largos de datos.
- ✓ Validar toda entrada con una lista “blanca” que contenga una lista de caracteres aceptados, siempre que sea posible.
- ✓ Comprobar si hay bytes nulos (%00).
- ✓ Comprobar si hay caracteres de nueva línea (%0d, %0a, \r, \n).
- ✓ Comprobar si hay caracteres de alteraciones de ruta “punto, punto, barra (../ o ..\). En los casos en que se soportan sets de caracteres UTF-8 extendidos, implemente representaciones alternativas tales como: %c0%ae%c0%ae/ (utilice la canonicalización como forma de implementar la doble codificación u otras formas de ofuscación de ataques).

2. Codificación de salidas de datos

- ✓ Realizar toda la codificación en un ambiente seguro (por ejemplo: el servidor).
- ✓ Utilizar una rutina probada y estándar para cada tipo de codificación de salida.
- ✓ Contextualizar la codificación de salida de todos los datos devueltos por el cliente que se originen desde fuera de la frontera de confianza de la aplicación. La codificación de entidades HTML es un ejemplo, aunque no sea suficiente en todos los casos.
- ✓ Codificar todos los caracteres salvo que sean reconocidos como seguros por el interpretador al que están destinados.
- ✓ Sanitizar (manejo de información confidencial) contextualmente todas las salidas de datos no confiables hacia consultas SQL, XML y LDAP.
- ✓ Sanitizar todas las salidas de datos no confiables hacia un comando del sistema operativo.

3. Administración de autenticación y contraseñas

- ✓ Requerir autenticación para todos los recursos y páginas excepto aquellas específicamente clasificadas como públicas.
- ✓ Todos los controles de autenticación deben ser efectuados en un sistema en el cual se confíe. (por ejemplo: el servidor).
- ✓ Establecer y utilizar servicios de autenticación estándares y probados cuando sea posible.
- ✓ Utilizar una implementación centralizada para todos los controles de autenticación, incluyendo librerías que llamen a servicios externos de autenticación.
- ✓ Segregar la lógica de la autenticación del recurso solicitado y utilizar redirección desde y hacia el control centralizado de autenticación.

- ✓ Todos los controles de autenticación deben fallar de una forma segura.
- ✓ Todas las funciones administrativas y de administración de cuentas deben ser al menos tan seguras como el mecanismo primario de autenticación.
- ✓ Si la aplicación administra un almacenamiento de credenciales, se debe asegurar que únicamente se almacena el hash con sal (salty hash) de las contraseñas y que el archivo/tabla que guarda las contraseñas y claves solo puede ser escrito por la aplicación (si es posible, no utilizar el algoritmo de hash MD5).
- ✓ El hash de las contraseñas debe ser implementado en un sistema en el cual se confíe (por ejemplo: el servidor).
- ✓ Validar los datos de autenticación únicamente luego de haber completado todos los datos de entrada, especialmente en implementaciones de autenticación secuencial.
- ✓ Las respuestas a los fallos en la autenticación no deben indicar cual parte de la autenticación fue incorrecta. A modo de ejemplo, en lugar de “usuario invalido” o “contraseña invalida”, utilizar “usuario y/o contraseña inválidos” en ambos casos. Las repuestas a los errores deben ser idénticas tanto a nivel de lo desplegado como a nivel de código fuente.
- ✓ Utilizar autenticación para conexiones a sistemas externos que involucren información o funciones sensibles.
- ✓ Las credenciales de autenticación para acceder a servicios externos a la aplicación deben ser encriptados y almacenados en ubicaciones protegidas en un sistema en el cual se confíe (ejemplo: el servidor). El código fuente NO es una ubicación segura.
- ✓ Utilizar únicamente conexiones encriptadas o datos encriptados para el envío de contraseñas que no sean temporales (por ejemplo: correo encriptado). Contraseñas temporales como aquellas asociadas con reseteos por correo electrónico, pueden ser una excepción.
- ✓ Hacer cumplir, por medio de una política o regulación, los requerimientos de complejidad de la contraseña. Las credenciales de autenticación deben ser suficientes como para resistir aquellos ataques típicos de las amenazas en el entorno del sistema.
- ✓ Hacer cumplir, por medio de una política o regulación, los requerimientos de longitud de la contraseña. Comúnmente se utilizan ocho caracteres, pero dieciséis es mejor.
- ✓ No se debe desplegar en la pantalla la contraseña ingresada. A modo de ejemplo, en formularios web, utilizar el tipo de entrada “clave” (input type “clave”).
- ✓ Deshabilitar las cuentas luego de un número establecido de intentos inválidos de ingreso al sistema. A modo de ejemplo, cinco intentos fallidos es lo común. La deshabilitación de la cuenta debe ser por un periodo de tiempo suficiente como para desalentar una inferencia de credenciales por fuerza bruta, pero no tan alto como para permitir un ataque de negación de servicio.
- ✓ El cambio y reseteo de contraseñas requieren los mismos niveles de control como aquellos asociados a la creación y autenticación de cuentas.
- ✓ Las preguntas para el reseteo de contraseñas deben contemplar un amplio rango de respuestas aleatorias. A modo de ejemplo: “¿libro favorito?” es una mala pregunta dado que “la biblia” es una respuesta muy común.
- ✓ Si se utiliza un reseteo por correo electrónico, únicamente enviar un link o contraseñas temporales a casillas previamente registradas en el sistema.
- ✓ Las contraseñas y links temporales deben tener un corto periodo de validez.
- ✓ Forzar el cambio de contraseñas temporales luego de su utilización.
- ✓ Notificar a los/las usuarios/as cada vez que se produce un reseteo de contraseña.
- ✓ Prevenir la reutilización de contraseñas.
- ✓ Las contraseñas deben tener al menos un día de antigüedad antes de poder ser cambiadas, a fin de evitar ataques de reutilización de contraseñas.
- ✓ Hacer cumplir, por medio de una política o regulación, los requerimientos de cambio de contraseña. Los sistemas críticos pueden requerir cambios más frecuentes que otros sistemas. El tiempo entre cada reseteo debe ser controlado administrativamente.
- ✓ Deshabilitar la funcionalidad de “recordar” campos de contraseñas.
- ✓ El último acceso (fallido o exitoso) debe ser reportado al/a la usuario/a en su siguiente acceso exitoso (buena práctica).
- ✓ Implementar un monitoreo para identificar ataques a múltiples cuentas utilizando la misma contraseña. Este patrón de ataque es utilizado parar superar bloqueos estándar cuando los nombres de usuario pueden ser obtenidos o adivinados de alguna forma.
- ✓ Cambiar todos los usuarios y contraseñas por defecto o deshabilitar las cuentas asociadas.
- ✓ Re autenticar usuarios antes de la realización de operaciones críticas.
- ✓ Utilizar autenticación multi-factor para las cuentas más sensibles o de mayor valor.
- ✓ Si se utiliza un código de una tercera parte para la autenticación, inspeccionarlo minuciosamente para asegurar que no se encuentre afectado por cualquier código malicioso.

4. Administración de sesiones

- ✓ Utilizar los controles del servidor o del framework para la administración de sesiones. La aplicación sólo debe reconocer estos identificadores como válidos.
- ✓ La creación de identificadores de sesión sólo debe ser realizada en un sistema en cual se confíe (por ejemplo: el servidor).
- ✓ Los controles de administración de sesiones deben utilizar algoritmos que generen identificadores suficientemente aleatorios.
- ✓ Definir el dominio y ruta para las cookies que contienen identificadores de sesión autenticados con un valor apropiadamente estricto para el sitio.
- ✓ La función de logout debe terminar completamente con la sesión o conexión asociada.
- ✓ La función de logout debe estar disponible en todas las páginas protegidas por autenticación.
- ✓ Establecer un tiempo de vida de la sesión lo más corto posible, balanceando los riesgos con los requerimientos del negocio. En la mayoría de los casos, nunca debería ser superior a varias horas.
- ✓ Deshabilitar logeos persistentes y efectuar finalizaciones periódicas de sesiones, incluso cuando la sesión se encuentra activa.
- ✓ Si una sesión fue establecida antes del login, cerrar dicha sesión y establecer una nueva luego de un login exitoso.
- ✓ Generar un nuevo identificador de sesión luego de cada re autenticación.
- ✓ No permitir logeos concurrentes con el mismo usuario.
- ✓ No exponer identificadores de sesión en URLs, mensajes de error ni logs. Los identificadores de sesión solo deben ser ubicados en la cabecera de la cookie HTTP. A modo de ejemplo, no transmitir el identificador de sesión como un parámetro GET.
- ✓ Proteger la información sobre las sesiones del lado del servidor implementando los controles de acceso apropiados.
- ✓ Generar un nuevo identificador de sesión y desactivar el anterior de forma periódica. De esta forma se pueden mitigar algunos escenarios de robo de sesiones donde el identificador se compromete.
- ✓ Generar un nuevo identificador de sesión si la seguridad cambia de HTTP a HTTPS, como puede suceder durante la autenticación. Dentro de la aplicación es recomendable usar siempre HTTPS en lugar de cambiar entre HTTP y HTTPS.
- ✓ Manejo de sesión complementario para operaciones sensible del lado del servidor, como pueden ser: gestión de cuentas o utilizando tokens o parámetros por sesión. Este método puede ser utilizado para prevenir ataques de Cross Site Request
- ✓ Forgery Attacks (Falsificación de petición en sitios cruzados, conocido por sus siglas CSRF).
- ✓ Manejo de sesión complementario para operaciones sensible o críticas utilizando tokens o parámetros por pedido (per request) en lugar de por sesión.
- ✓ Configurar el atributo “seguro” para las cookies transmitidas sobre una conexión TLS.
- ✓ Configurar las cookies con el atributo HttpOnly, salvo que se requiera específicamente scripts del lado del cliente en la aplicación, para leer o configurar una cookie.

5. Control de Acceso

- ✓ Utilizar un único componente para el chequeo de autorizaciones para todo el sitio. Esto incluye librerías que llamen a servicios de autorización externos.
- ✓ Los controles de acceso en caso de falla, deben actuar en forma segura.
- ✓ Denegar todos los accesos en caso de que la aplicación no pueda acceder a la información de configuración de seguridad.
- ✓ Requerir controles de autorización en cada solicitud o pedido, incluyendo aquellos creados por scripts en el servidor.
- ✓ Separar lógica privilegiada de otro código de la aplicación.
- ✓ Restringir acceso a ficheros u otros recursos, incluyendo aquellos fuera del control directo de la aplicación, únicamente a usuarios autorizados.
- ✓ Restringir el acceso a URLs protegidas, solo a usuarios autorizados.
- ✓ Restringir el acceso a funciones protegidas, solo a usuarios autorizados.
- ✓ Restringir las referencias directas a objetos, solo a usuarios autorizados.
- ✓ Restringir el acceso a servicios, solo a usuarios autorizados.
- ✓ Restringir el acceso a información de la aplicación, solo a usuarios autorizados.
- ✓ Restringir el acceso a usuario, atributos y política de información utilizada por los controles de acceso.

- ✓ Restringir el acceso a información relevante de la configuración, solo a usuarios autorizados.
- ✓ Las reglas de control de acceso implementadas del lado del servidor y en la capa de presentación, deben coincidir.
- ✓ Si existen datos de estado que deben ser guardados en el cliente, use cifrado y chequeos de integridad del lado del servidor para poder evaluar el estado.
- ✓ Requerir flujos de aplicación que cumplan con las reglas del negocio.
- ✓ Limitar el número de transacciones que un usuario común o un dispositivo puede desarrollar en un cierto período de tiempo.
- ✓ Utilizar el header "referer" solo como un chequeo complementario. Nunca debe ser utilizado como chequeo de autorización, ya que es posible modificarlo.
- ✓ Si sesiones de autenticación extensas son permitidas, periódicamente hay que re-validar la autorización de los usuarios y asegurar que sus privilegios no han sido modificados.
- ✓ Implementar auditorías de cuentas y requerir que se deshabiliten las contraseñas de las cuentas.
- ✓ La aplicación debe permitir deshabilitar y terminar cuentas una vez que se termina la autorización (Cambio de rol, estatus de empleo, etc).
- ✓ Cuentas de servicio o cuentas soportando conectividad deben tener el mínimo privilegio.
- ✓ Crear una política de control de acceso para documentar las reglas del negocio de la aplicación, los tipos de datos, criterios para autorización de acceso y los controles asociados para otorgarlos y controlarlos. Esto incluye la identificación de accesos requeridos tanto para los datos como para los sistemas.

6. Criptografía

- ✓ Todas las funciones de criptografía de la aplicación deben ser implementadas en sistemas de confiables (por ejemplo: el servidor).
- ✓ Proteger secretos maestros (master secrets) del acceso no autorizado.
- ✓ Los módulos de criptografía deberían en caso de falla, fallar en forma segura.
- ✓ Todos los números aleatorios, nombres aleatorios, GUIDs, y frases aleatorias, deberían generarse utilizando módulos aprobados para su generación.
- ✓ Los módulos criptográficos del sistema, deben cumplir con FIPS 140-2 o con su estándar equivalente.
- ✓ Establecer y utilizar una política y un proceso de cómo manejar las claves criptográficas.

7. Manejo de errores y Logs

- ✓ No difundir información sensible en respuestas de error, incluyendo detalles del sistema, identificadores de sesión o información de la cuenta.
- ✓ Utilizar manejadores de errores que no muestren información de debugging o de memoria.
- ✓ Implementar mensajes de error genéricos y utilizar páginas de error adaptadas.
- ✓ La aplicación debería manejar los errores de la aplicación y basarse en la configuración del servidor.
- ✓ Liberar espacio de memoria en cuanto una condición de error ocurra.
- ✓ La lógica para la gestión de errores debe estar asociada a que los controles de seguridad no permitirán acceso por defecto.
- ✓ Todos los controles de logging deben estar implementados en sistemas confiables.
- ✓ El logging de controles de acceso debe incluir tanto los casos de éxito como de falla.
- ✓ Asegurar que los datos del registro de log contengan información importante.
- ✓ Asegurar que los logs de entrada que incluyen información no segura, no serán ejecutados en interfaces o aplicativos.
- ✓ Restringir el acceso a los logs, solo a personal autorizado.
- ✓ Utilizar una rutina centralizada para todas las operaciones de logging.
- ✓ No guardar información sensible en logs, incluyendo detalles innecesarios del sistema.
- ✓ Asegurar que existen mecanismos para conducir un análisis de los logs.
- ✓ Registrar en un log todas las fallas de validación.
- ✓ Registrar en un log todos los intentos de autenticación, en particular los fallidos.
- ✓ Registrar en un log todas las fallas en los controles de acceso.
- ✓ Registrar en un log todos los eventos de intento de evasión de controles, incluyendo cambios en el estado de la información no esperados.
- ✓ Registrar en un log todos los intentos de conexión con tokens inválidos o vencidos.
- ✓ Registrar en un log todas las excepciones del sistema.
- ✓ Registrar en un log todas las funciones administrativas, incluyendo cambios en la configuración de seguridad.

- ✓ Registrar en un log, todas las fallas de conexión de TLS.
- ✓ Registrar en un log las fallas de los módulos criptográficos.
- ✓ Utilizar una función de hash para validar la integridad de los logs.

8. Protección de Datos

- ✓ Implementar privilegios mínimos, restrinja a los usuarios solo a la funcionalidad, los datos y la información del sistema necesarios para realizar sus tareas.
- ✓ Proteger todas las copias en caché o temporales de datos confidenciales almacenados en el servidor del acceso no autorizado y eliminar esos archivos de trabajo temporales tan pronto como ya no sean necesarios.
- ✓ Cifrar información almacenada altamente confidencial, como datos de verificación de autenticación.
- ✓ Proteger el código fuente del lado del servidor para que no sea descargado por un usuario.
- ✓ No almacenar contraseñas, cadenas de conexión u otra información confidencial en texto sin cifrar o de ninguna manera segura no criptográfica en el lado del/de la cliente. Esto incluye la incrustación en formatos inseguros como: MS viewstate, Adobe flash o código compilado.
- ✓ Eliminar los comentarios en el código de producción accesible para el/la usuario/a, que puedan revelar el sistema de back-end u otros como información sensible.
- ✓ Eliminar la documentación innecesaria de la aplicación y el sistema, ya que puede revelar información útil para atacantes.
- ✓ No incluir información confidencial en los parámetros de solicitud HTTP GET.
- ✓ Deshabilitar las funciones de autocompletar en los formularios que se espera que contengan información confidencial, incluidos autenticación.
- ✓ Deshabilitar el almacenamiento en caché del lado del/de la cliente en páginas que contengan información confidencial. Cache-Control: sin almacenamiento, puede usarse junto con el control de encabezado HTTP "Pragma: no-cache", que es menos efectivo, pero HTTP/1.0 es compatible con versiones anteriores.
- ✓ La aplicación debe permitir la eliminación de datos confidenciales cuando ya no se necesiten. Por ejemplo, información personal o ciertos datos financieros.
- ✓ Implementar controles de acceso apropiados para los datos confidenciales almacenados en el servidor. Esto incluye datos almacenado en caché, archivos temporales y datos a los que solo deben tener acceso usuarios específicos del sistema

9. Seguridad de las comunicaciones

- ✓ Utilizar el protocolo TLS para conexiones a sistemas externos que involucren información o funciones confidenciales. Emplear una versión vigente y confiable de TLS (1.3 como recomendación).
- ✓ Implementar encriptación para la transmisión de toda la información sensible. Esto debe incluir TLS para proteger la conexión y puede complementarse con cifrado discreto de archivos confidenciales o conexiones no basadas en HTTP.
- ✓ Los certificados TLS deben ser válidos y tener el nombre de dominio correcto, no deben estar vencidos.
- ✓ Las conexiones TLS fallidas no deberían recurrir a una conexión insegura.
- ✓ Utilizar conexiones TLS para todo el contenido que requiera acceso autenticado y para toda la información.
- ✓ Utilizar una única implementación de TLS estándar que esté configurada adecuadamente.
- ✓ Especificar codificaciones de caracteres para todas las conexiones.

10. **Configuración del sistema**

- ✓ Asegurarse de que los servidores, los frameworks y los componentes del sistema estén ejecutando la última versión aprobada.
- ✓ Asegurarse de que los servidores, los frameworks y los componentes del sistema tengan todos los parches emitidos para la versión en uso.
- ✓ Desactivar las listas de directorios.
- ✓ Restringir el servidor web, el proceso y las cuentas de servicio a los mínimos privilegios posibles.
- ✓ Cuando ocurran excepciones, debe fallar de forma segura.
- ✓ Eliminar todas las funciones y archivos innecesarios.
- ✓ Eliminar el código de las pruebas o cualquier funcionalidad no destinada a la producción, antes de la implementación.

- ✓ Definir que métodos HTTP, GET o POST, admitirá la aplicación y si se manejará de manera diferente en cada página de la aplicación.
- ✓ Deshabilitar los métodos HTTP innecesarios, como las extensiones WebDAV. Si se requiere un método HTTP extendido que admita el manejo de archivos, utilizar un mecanismo de autenticación bien examinado.
- ✓ Si el servidor web maneja HTTP 1.0 y 1.1, asegurarse de que ambos estén configurados de manera similar, y comprender cualquier diferencia que pueda existir (por ejemplo, manejo de métodos HTTP extendidos).
- ✓ Eliminar la información innecesaria de los encabezados de respuesta HTTP relacionados con el sistema operativo, la versión del servidor web y frameworks.
- ✓ La configuración de seguridad para la aplicación debe generarse en formato legible para cualquier lector y servir de apoyo en las auditorías.
- ✓ Implementar un sistema de control de cambios de software para administrar y **registrar los cambios del código tanto en desarrollo como en producción.**

11. Seguridad en base de datos

- ✓ Usar consultas fuertemente parametrizadas (Parameterized Queries).
- ✓ Utilizar la validación de entrada y la codificación de salida y asegurarse de abordar los metacaracteres.
- ✓ Asegurarse de que las variables estén fuertemente tipificadas.
- ✓ La aplicación debe usar el nivel de privilegios más bajo posible al acceder a la base de datos.
- ✓ Usar credenciales seguras para acceder a la base de datos.
- ✓ Las cadenas de conexión no deben codificarse de forma rígida dentro de la aplicación.
- ✓ Las cadenas de conexión deben ser almacenados en un archivo de configuración separado en un sistema confiable y deben estar encriptados.
- ✓ Usar procedimientos almacenados para abstraer el acceso a los datos y permitir la eliminación de permisos a la base tablas en la base de datos.
- ✓ Cerrar la conexión lo antes posible.
- ✓ Desactivar todas las funciones innecesarias de la base de datos [por ejemplo: procedimientos o servicios almacenados innecesarios, paquetes, instalar solo el conjunto mínimo de características y opciones requeridas (reducción del área de superficie)].
- ✓ Deshabilitar cualquier cuenta predeterminada que no sea necesaria para cumplir con los requisitos comerciales.
- ✓ La aplicación debe conectarse a la base de datos con diferentes credenciales para cada distinción de confianza (por ejemplo, usuario, usuario de solo lectura, invitado, administradores, etc.).

12. Gestión de archivos

- ✓ Requerir autenticación antes de permitir que se cargue un archivo.
- ✓ Limitar el tipo de archivos que se pueden cargar a sólo aquellos tipos que se necesitan para fines autorizados.
- ✓ Validar que los archivos cargados sean del tipo esperado comprobando los encabezados de los archivos. Verificar el tipo de archivo, solo por extensión no es suficiente.
- ✓ No guardar archivos en el mismo contexto web de la aplicación. Los archivos deben ir al servidor de contenido o a la base de datos.
- ✓ Impedir o restringir la carga de cualquier archivo que pueda ser interpretado por el servidor web.
- ✓ Desactivar los privilegios de ejecución en los directorios de carga de archivos.
- ✓ Cuando se haga referencia a archivos existentes, usar una lista blanca de nombres y tipos de archivos permitidos. Validar el valor del parámetro que se pasa y si no coincide con uno de los valores esperados, rechazarlo o usar un valor de archivo predeterminado codificado para el contenido.
- ✓ Nunca enviar la ruta absoluta del archivo al/a la cliente.
- ✓ Asegurarse de que los archivos y recursos de la aplicación sean sólo de lectura.

13. Gestión de la memoria

- ✓ Utilizar control de entrada y salida para datos no confiables.
- ✓ Comprobar los límites del buffer si llama a la función en un bucle y asegurarse de que no haya peligro de escribir más allá del espacio asignado.
- ✓ Truncar todas las cadenas de entrada a una longitud razonable antes de pasarlas a las funciones

de copia y concatenación.
- ✓ Cerrar específicamente los recursos, no confiar en la recolección de basura (por ejemplo: objetos de conexión, identificadores de archivos, etc.).
- ✓ Evitar el uso de funciones vulnerables conocidas (por ejemplo: printf, strcat, strcpy, etc.).
- ✓ Memoria asignada correctamente, se libera al finalizar las funciones y en todos los puntos de salida.

14. Prácticas generales de codificación

- ✓ Usar código administrado probado y aprobado en lugar de crear código nuevo no administrado para tareas comunes.
- ✓ Utilizar API integradas de tareas específicas para realizar tareas del sistema operativo.
- ✓ No permitir que la aplicación emita comandos directamente al sistema operativo, especialmente mediante el uso de Shell de comandos iniciados por la aplicación.
- ✓ Usar sumas de verificación y hashes para verificar la integridad del código interpretado, las librerías, los ejecutables y el archivo de configuración.
- ✓ Proteger las variables y los recursos compartidos de acceso simultáneo inapropiado.
- ✓ Inicializar explícitamente todas sus variables y otros almacenes de datos, ya sea durante la declaración o justo antes del primer uso.
- ✓ En los casos en que la aplicación deba ejecutarse con privilegios elevados, aumentar los privilegios lo más tarde posible, y dejarlos el menor tiempo posible.
- ✓ No pasar datos proporcionados por el/la usuario/a a ninguna función de ejecución dinámica.
- ✓ Impedir que los/las usuarios/as generen código nuevo o modifiquen el código existente.
- ✓ Revisar todas las aplicaciones secundarias, código de terceros y librerías para determinar la necesidad del negocio y validar la funcionalidad segura, ya que pueden introducir nuevas vulnerabilidades.
- ✓ Implementar actualizaciones seguras. Si la aplicación va a utilizar actualizaciones automáticas, utilizar firmas criptográficas para su código y asegurarse de que los/las clientes de descarga verifiquen esas firmas. Usar canales encriptados para transferir el código desde el servidor host.

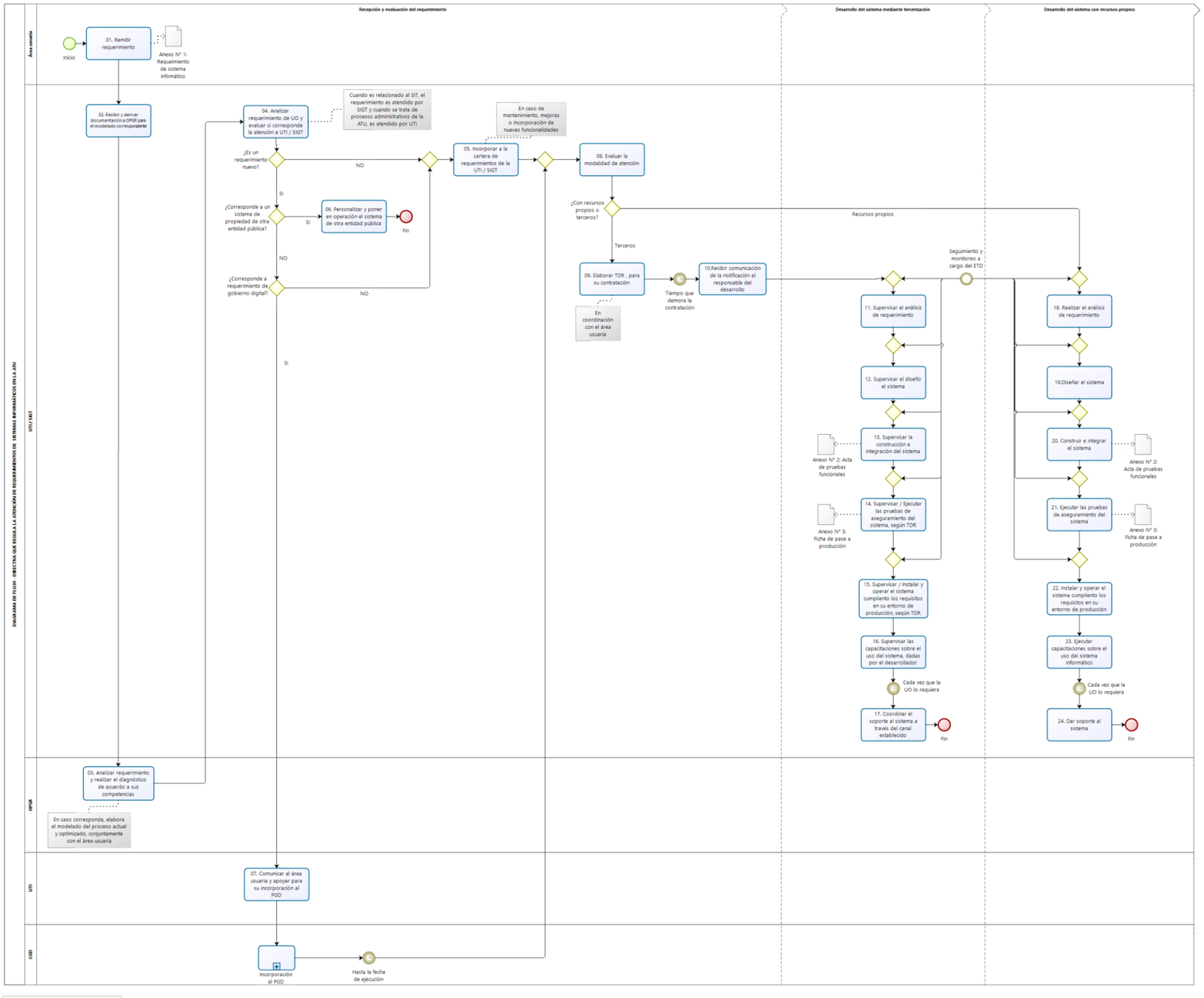
ANEXO 6:

PORTAFOLIO PLAN DE GOBIERNO DIGITAL – FICHA DE PROYECTO

Nombre del Proyecto **PGD-X**	**NOMBRE DEL PROYECTO**			
Área responsable	*<Nombre del área responsable de la implementación del proyecto>*			
Beneficiarios	*<Especificar el(los) beneficiario(s)>*			
Tipo de Proyecto (Marque "X")		De cara al ciudadano o administrado		Mejora de la Gestión Interna
Problemas para solucionar / Brecha a atender	*<Especificar la situación actual, los problemas de no contar con el proyecto, así como la brecha que se cerraría luego de ejecutar el proyecto>*			
Descripción del Proyecto	*<Especificar las funcionalidades de la solución>*			
Beneficio a obtener	*<Listar los beneficios a obtener con la implementación del proyecto>*			
Riesgos	*<Listar los riesgos a considerar en la implementación del proyecto>*			
Costo estimado	*<S/ XXX,XXX.00* *Especificar si el costo de implementación del proyecto corresponde a un desarrollo o adquisición…Si el monto estimado está considerado en el cuadro de necesidades multianual o plan operativo de la unidad orgánica>.*			
Plazo de implementación	**Fecha de Inicio:** *<Mes Año>* **Duración:** *<Cantidad de meses>* *<Considerar que la fecha de inicio debe ser posterior a la aprobación del CGD (Comité del gobierno Digital de la ATU) y la asignación presupuestal correspondiente>*			

ANEXO 7:

DIAGRAMA DE FLUJO - DIRECTIVA QUE REGULA LA ATENCIÓN E IMPLEMENTACION DE SISTEMAS INFORMÁTICOS EN LA ATU



CGD: Comité de Gobierno Digital
ETD: Equipo Tecnico de Desarrollo
OPGR: Oficina de Procesos y Gestión de Riesgos
PGD: Plan de Gobierno Digital
SIT: Sistema Integrado de Transporte de Lima y Callao
SIGT: Subdirección de Integración y Gestión Tecnológica
TDR: Términos de referencia
UTI: Unidad de Tecnologías de la Información