Proyecto:

Documento de Arquitectura del Sistema  
(SAD)

**Software Architecture Document**

**IEEE-1471-2000**

Equipo de Trabajo:

Diego Madariaga

Bastián Lillo

Celso Lagos

Diego Silva

Jordán Donoso

|  |
| --- |
| HISTORIAL DE VERSIONES |

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 06-10-2016 | 1.0 | Creación del documento | Equipo |
| 19-10-2016 | 1.1 | Generación de vistas de arquitectura | Diego Madariaga |
| 24-10-2016 | 1.2 | Diseño del Modelo Relacional | Jordán Donoso |
| 26-10-2016 | 1.7 | Término de Informe. | Diego Madariaga |
| 31-10-2016 | 1.8 | Incorporación de framework | Celso Lagos |
| 01-11-2016 | 1.9 | Control de calidad QA | Jordán Donoso |
|  |  |  |  |
|  |  |  |  |

|  |
| --- |
| TABLA DE CONTENIDO |

Contenido

[TABLA DE CONTENIDO 3](#_Toc466028426)

[LISTADO DE FIGURAS 4](#_Toc466028427)

[LISTADO DE TABLAS 5](#_Toc466028428)

[SECCION 1: DESCRIPCIÓN DEL DOCUMENTO 6](#_Toc466028429)

[1.1 Propósito y Alcance 6](#_Toc466028430)

[1.2 Convenciones 7](#_Toc466028431)

[1.3 Terminología y Definiciones 7](#_Toc466028432)

[1.4 Documentos Relevantes 7](#_Toc466028433)

[SECCION 2: CONCEPTUAL FRAMEWORK 8](#_Toc466028434)

[2.1 Architectural description in context 8](#_Toc466028435)

[2.3 Architectural activities in the life cycle 8](#_Toc466028436)

[Característica de AUP 8](#_Toc466028437)

[2.4 Stakeholders 11](#_Toc466028438)

[SECCION 3: ARCHITECTURAL DESCRIPTION PRACTICES 13](#_Toc466028439)

[3.1 Architectural documentation 13](#_Toc466028440)

[3.2 Identification of stakeholders and concerns 14](#_Toc466028441)

[3.3 Selection of architectural viewpoints 14](#_Toc466028442)

[3.4 Architectural views 14](#_Toc466028443)

[3.4.1 Architectural views ESCENARIOS. 15](#_Toc466028444)

[3.4.2 Architectural views LÓGICA. 16](#_Toc466028445)

[3.4.3 Architectural views DESARROLLO. 17](#_Toc466028446)

[3.4.4 Architectural views FÍSICA 18](#_Toc466028447)

[3.4.5 Architectural views PROCESOS. 18](#_Toc466028448)

[3.4.6 Architectural views DATOS. 20](#_Toc466028449)

[3.5 Consistency among architectural views 21](#_Toc466028450)

[SECCION 4: REQUERIMIENTOS DE CALIDAD o ARCHITECTURAL RATIONALE 24](#_Toc466028451)

[4.1 Árbol de Utilidad 24](#_Toc466028452)

[4.2 Escenarios de Calidad Priorizados 26](#_Toc466028453)

[1 Control y libertad del usuario 30](#_Toc466028454)

[2 Evitar errores 30](#_Toc466028455)

[3 Diseño minimalista 30](#_Toc466028456)

[4 Reconocer, diagnosticar y recuperarse de los errores 31](#_Toc466028457)

[5 Correspondencia entre el sistema y el mundo real 31](#_Toc466028458)

[SECCION 6: SIMBOLOGÍA 32](#_Toc466028459)

[SECCION 6: SIMBOLOGÍA 34](#_Toc466028460)

[SECCION 7: Framework 34](#_Toc466028461)

[Framework Swing 34](#_Toc466028462)

|  |
| --- |
| LISTADO DE FIGURAS |

[Ilustración 1 Escenarios 14](#_Toc465370773)

[Ilustración 2 Lógica 15](#_Toc465370774)

[Ilustración 3 Desarrollo 16](#_Toc465370775)

[Ilustración 4 Física 17](#_Toc465370776)

[Ilustración 5 Procesos 17](#_Toc465370777)

[Ilustración 6 Datos 19](#_Toc465370778)

|  |
| --- |
| LISTADO DE TABLAS |

[Tabla 1 Listado de los Stakeholders 10](#_Toc465370619)

[Tabla 2 Stakeholders y Expectativas 10](#_Toc465370620)

[Tabla 3 Stakeholders y Vistas 13](#_Toc465370621)

[Tabla 4 Modulos 20](#_Toc465370622)

[Tabla 5 conectores 22](#_Toc465370623)

|  |
| --- |
| SECCION 1: DESCRIPCIÓN DEL DOCUMENTO |

|  |
| --- |
| 1.1 Propósito y Alcance |

(Breve descripción de la organización y los usuarios a los que está dirigido este documento y objetivos del proyecto)

Este documento está dirigido a todos los involucrados en el proyecto “Escuela de vuelo Halcón”, empresa que necesita automatizar sus procesos, para así entregar un servicio de mejor calidad a sus clientes.

Se especificarán las problemáticas que sufren hoy en día, los cuales son: no tener automatizado el control de los mantenimientos, de los vuelos, gestión de los usuarios del sistema.

En este documento, también se plantean las soluciones que proponemos como empresa, las cuales se basan en el análisis de los problemas de nuestro cliente. También se especifican los diagramas de las soluciones que planteadas, los cuales son: diagrama de casos de uso de alto nivel, casos de uso de negocio, diagrama de clases, etc.

Para llevar a cabo la solución de nuestro cliente, se han realizado una serie de diagramas basados en el modelo 4+1 de Kruchten, el cual trata de cuatro vistas (Lógica, Física, Despliegue, Procesos), los cuales reflejan lo que será el gran sistema que ayudará a mejorar los procesos de la empresa LAN SUR

|  |
| --- |
| 1.2 Convenciones |

(Descripción de las notaciones y símbolos utilizados en este documento)

Estándar UML usado en StarUML 6.0

BPMN usado en Bizagi.

Modelo de Datos en Oracle DataBase.

|  |
| --- |
| 1.3 Terminología y Definiciones |

(Descripción de los términos utilizados en el documento y parte del dominio y contexto del problema)

**Aeronave**: Una aeronave es cualquier vehículo capaz de navegar por el aire, o, en general, por la atmósfera de un planeta

**DGAC**: Dirección General de Aeronáutica Civil, ente encargado de Normar y fiscalizar la actividad aérea que se desarrolla dentro del espacio aéreo controlado por Chile y aquella que ejecutan en el extranjero empresas aéreas nacionales: desarrollar la infraestructura aeronáutica en el ámbito de su competencia y prestar servicios de excelencia de navegación aérea, meteorología, aeroportuarios y seguridad operacional, con el propósito de garantizar la operación del Sistema Aeronáutico en forma segura y eficiente.

**Aeronavegabilidad**: Característica o condiciones que deben reunir las aeronaves para realizar en forma segura y satisfactoria los vuelos o maniobras para las que han sido autorizadas.

**Medicina aeroespacial:** La medicina aeroespacial o medicina aeronáutica es la especialidad de la medicina preventiva que estudia las enfermedades y trastornos del organismo humano asociados con el vuelo.

|  |
| --- |
| 1.4 Documentos Relevantes |

(Listado de documentos relevantes, utilizados durante el desarrollo de la arquitectura, anexar el documento corregido de elicitación y especificación de Requisitos (830 extendida))

Anexo ERS.doc

|  |
| --- |
| SECCION 2: CONCEPTUAL FRAMEWORK |

|  |
| --- |
| 2.1 Architectural description in context |

(Breve descripción de modelo 4+1 DE Kruchten.)

El modelo 4+1 describe la arquitectura del software usando cinco vistas concurrentes.

• La vista lógica describe el modelo de objetos del diseño cuando se usa un método de diseño orientado a

objetos. Para diseñar una aplicación muy orientada a los datos, se puede usar un enfoque alternativo

para desarrollar algún otro tipo de vista lógica, tal como diagramas de entidad-relación.

• La vista de procesos describe los aspectos de concurrencia y sincronización del diseño.

• La vista física describe el mapeo del software en el hardware y refleja los aspectos de distribución.

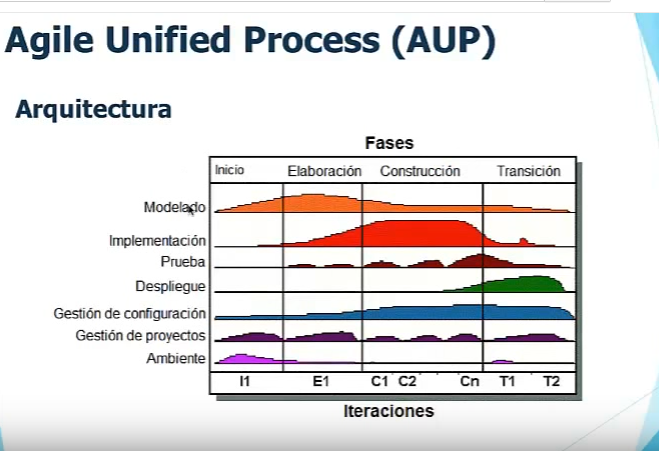
• La vista de desarrollo describe la organización estática del software en su ambiente de desarrollo.

Diagramas relacionados: diagrama de clases, diagrama de componentes, diagrama lógico, diagrama de procesos, diagrama de Secuencia, diagrama de Colaboración, diagrama de Estado y diagrama de Interacción.

|  |
| --- |
| 2.3 Architectural activities in the life cycle |

(Describir un ciclo de vida propuesto para el desarrollo del proyecto.)

**Agile Unified Process**, en español Proceso Unificado Ágil de Scott Ambler o (AUP) en inglés, es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP.



## Característica de AUP

- Versión simplificada de la metodología RUP.

- Abarca siete flujos de trabajos, cuatro ingenieriles y tres de apoyo: Modelado, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de Proyectos y Ambiente.

- Dispone de cuatro fases igual que RUP: Incepción o Creación, Elaboración, Construcción y Transición.

**Descripción de los flujos de trabajo ingenieriles**

* El **Modelado** es el flujo de trabajo que tiene el objetivo de entender el negocio de la organización, el problema de dominio que se aborda en el proyecto y determinar una solución viable para resolver el problema de dominio.
* El flujo de trabajo **Implementación** tiene como objetivo transformar su (s) modelo (s) en código ejecutable y realizar un nivel básico de las pruebas, en particular, la unidad de pruebas.
* El flujo de trabajo de **Prueba** tiene como objetivo realizar una evaluación objetiva para garantizar la calidad. Esto incluye la búsqueda de defectos, validar que el sistema funciona tal como está establecido, verificando que se cumplan los requerimientos.
* Por último, dentro de los flujos de trabajo ingenieriles se tiene el **Despliegue**, cuyo objetivo es el plan para la prestación del sistema y la ejecución de dicho plan, para que el sistema quede a disposición de los usuarios finales.

**Descripción de las fases del ciclo de desarrollo**

* Incepción: identificación del alcance y dimensión del proyecto, propuesta de la arquitectura y del presupuesto del cliente.
* Elaboración: Confirmación de la idoneidad de la arquitectura.
* Construcción: Desarrollo incremental del sistema, siguiendo las prioridades funcionales de los implicados.
* Transición: Validación e implantación del sistema.

**Técnicas ágiles que aplica AUP**

* Desarrollo Dirigido por Pruebas (Test Driven Development - TDD)
* Modelado Ágil
* Gestión de Cambios Ágil
* Refactorización de Base de Datos para mejorar la productividad.

**Principios en los que se basa AUP**

**Simplicidad**: Todo se describe concisamente utilizando poca documentación, no miles de ellas.

**Agilidad**: El ajuste a los valores y principios de *La Alianza Ágil*.

**Centrarse en actividades de alto valor**: La atención se centra en las actividades que en realidad lo requieren, no en todo el proyecto.

**Herramienta de la independencia**: Usted puede usar cualquier conjunto de herramientas que desea con el AUP. Se sugiere utilizar las herramientas más adecuadas para el trabajo, que a menudo son las herramientas simples o incluso herramientas de código abierto.

**Usted querrá adaptar este producto para satisfacer sus propias necesidades**: La metodología AUP es un producto de fácil uso utilizando cualquier herramienta. No es necesario comprar una herramienta especial, o tomar un curso, para adaptar esta metodología.

**Principios de La *Alianza Ágil***

1. La mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptar que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregar software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajan juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Fuente: https://www.ecured.cu/Agile\_Unified\_Process

|  |
| --- |
| 2.4 Stakeholders |

Tabla 1 Listado de los Stakeholders

|  |  |
| --- | --- |
| Stakeholder | Descripción |
| Administrador | persona que administra la creación y gestión de los demás usuarios en el sistema. |
| Operador | persona que administra: vuelos, personal de la escuela, aeronaves, componentes y los mantenimientos de estos 2 últimos. |
| Piloto | miembro de la tripulación que opera los controles de la aeronave |
| Alumno | persona que está en proceso de formación como piloto |
| Consultor | Personal fiscalizador de la DGAC |
| tripulación | persona que viaja en calidad de tripulante en un vuelo |
| Presidente de la escuela | encargado de la gestión general de la escuela de vuelo Escuadrilla Halcón |
|  |  |

Tabla 2 Stakeholders y Expectativas

|  |  |
| --- | --- |
| Stakeholder | Expectativa |
| Administrador | Facilitar la gestión de los usuarios y hacerlo de forma segura. |
| Operador | Facilitar la gestión de: los pilotos y sus licencias, de aeronaves, componentes y los mantenimientos asociados a estos. |
| Piloto | Reducir los tiempos de la visión de las horas de vuelo y los mantenimientos de una aeronave |
| Alumno | Reducir los tiempos de la visión de las horas de vuelo y los mantenimientos de una aeronave |
| Consultor | Agilizar las fiscalizaciones realizadas a las naves y sus cumplimientos |
| tripulante | Mayor seguridad al viajar en aeronaves con mantenimientos controlados de mejor manera |
| Presidente de la escuela | Disminución de las probabilidades de un accidente o multa por parte de la DGAC, provocado por algún control del mantenimiento mal realizado. |
|  |  |

|  |
| --- |
| SECCION 3: ARCHITECTURAL DESCRIPTION PRACTICES |

|  |
| --- |
| 3.1 Architectural documentation |

(Esta sección describe en qué documentos se basa la arquitectura 4+1)

Para diseñar el sistema, decidimos como equipo que la manera más apropiada es ocupar las vistas 4+1, las cuales consisten en diferentes diagramas, los cuales son:

* **Lógica**: Diagrama que representa a clases, las cuales contienen atributos y operaciones y a sus relaciones.
* **Física**: Diagrama que representa la configuración física del proyecto, las cuales representan tanto partes del software como hardware.
* **Procesos**: Diagrama que representa los procesos que se tienen que llevar a cabo para satisfacer un requerimiento. En nuestro caso ocupamos el diagrama BPMN.
* **Despliegue**: Diagrama que divide el sistema en pequeños paquetes y componentes. Para llevar a cabo este proyecto, usamos el diagrama de componentes.
* **Escenarios**: Diagrama que representa lo que hará cada usuario en el sistema.

Adicionalmente, insertamos otra vista:

* **Datos**: Diagrama relacional el cual representa las tablas y sus relaciones de la base de datos que ocupará el sistema.

|  |
| --- |
| 3.2 Identification of stakeholders and concerns |

(Esta sección describe a los interesados, relacionados con sus expectativas en las distintas vistas de la arquitectura)

Tabla 3 Stakeholders y Vistas

|  |  |  |  |
| --- | --- | --- | --- |
| **Stakeholder** | **descripción** | **escenario** | **Vistas** |
| Equipo de Programación | Equipo encargado de la programación del sistema |  | Lógica, desarrollo, datos |
| Analista de Requisitos | Encargado de la especificación de los requisitos del sistema |  | Escenarios, procesos, bpmn |
| SCM | Encargado de la configuración de ambiente para el sistema |  | Física |
| Cliente |  |  | Escenarios |

|  |
| --- |
| 3.3 Selection of architectural viewpoints |

|  |  |
| --- | --- |
| Vistas | UML |
| Escenarios | Casos de uso |
| Lógica | Clases |
| Desarrollo | Componentes |
| Física | Despliegue |
| Procesos | BPMN |
| Datos | Modelo Relacional |

|  |
| --- |
| 3.4 Architectural views |

(Esta sección se desarrollan todas las vistas del modelo 4+1)

### 3.4.1 Architectural views ESCENARIOS.

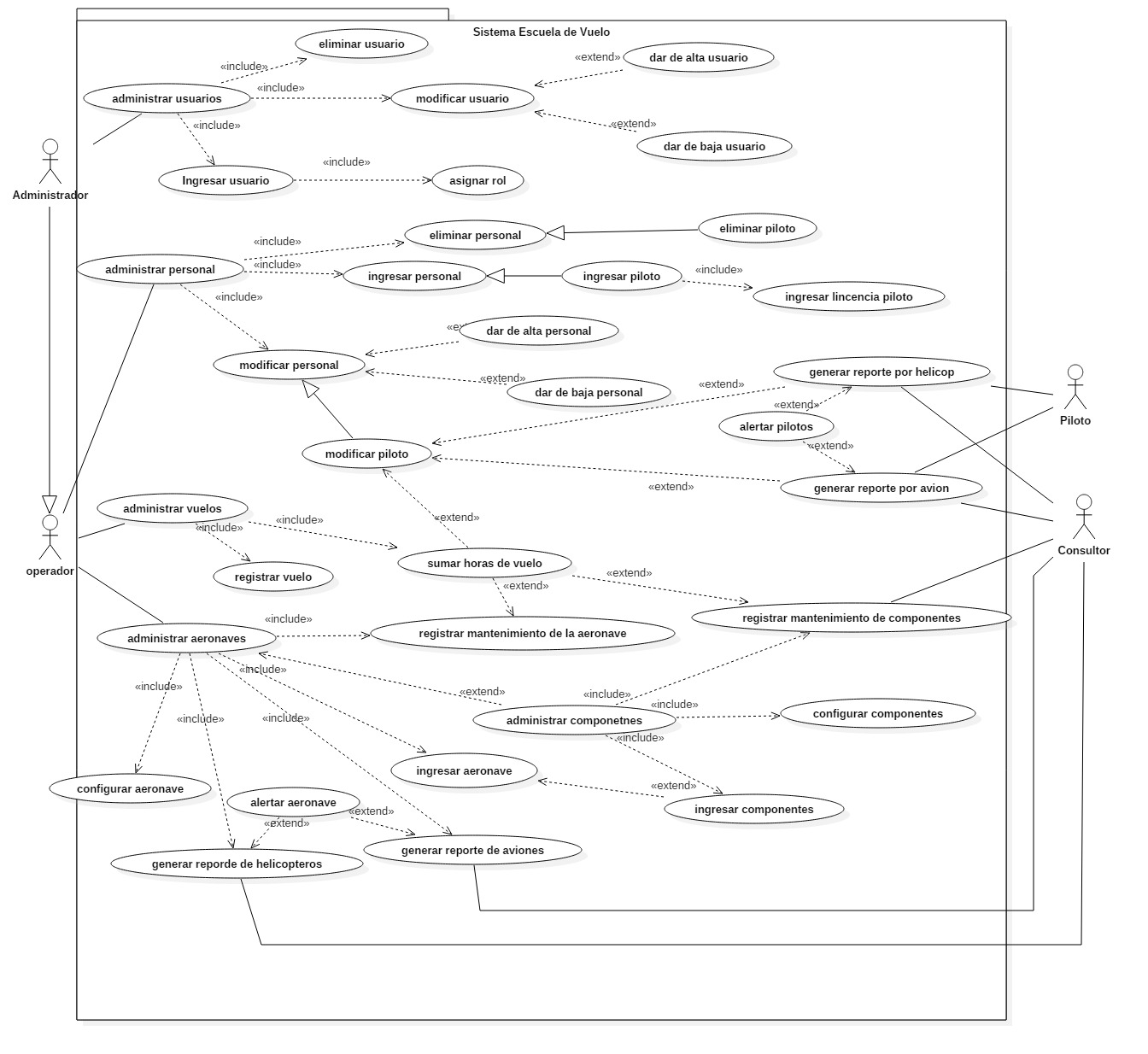
****

Ilustración 1 Escenarios

### 3.4.2 Architectural views LÓGICA.

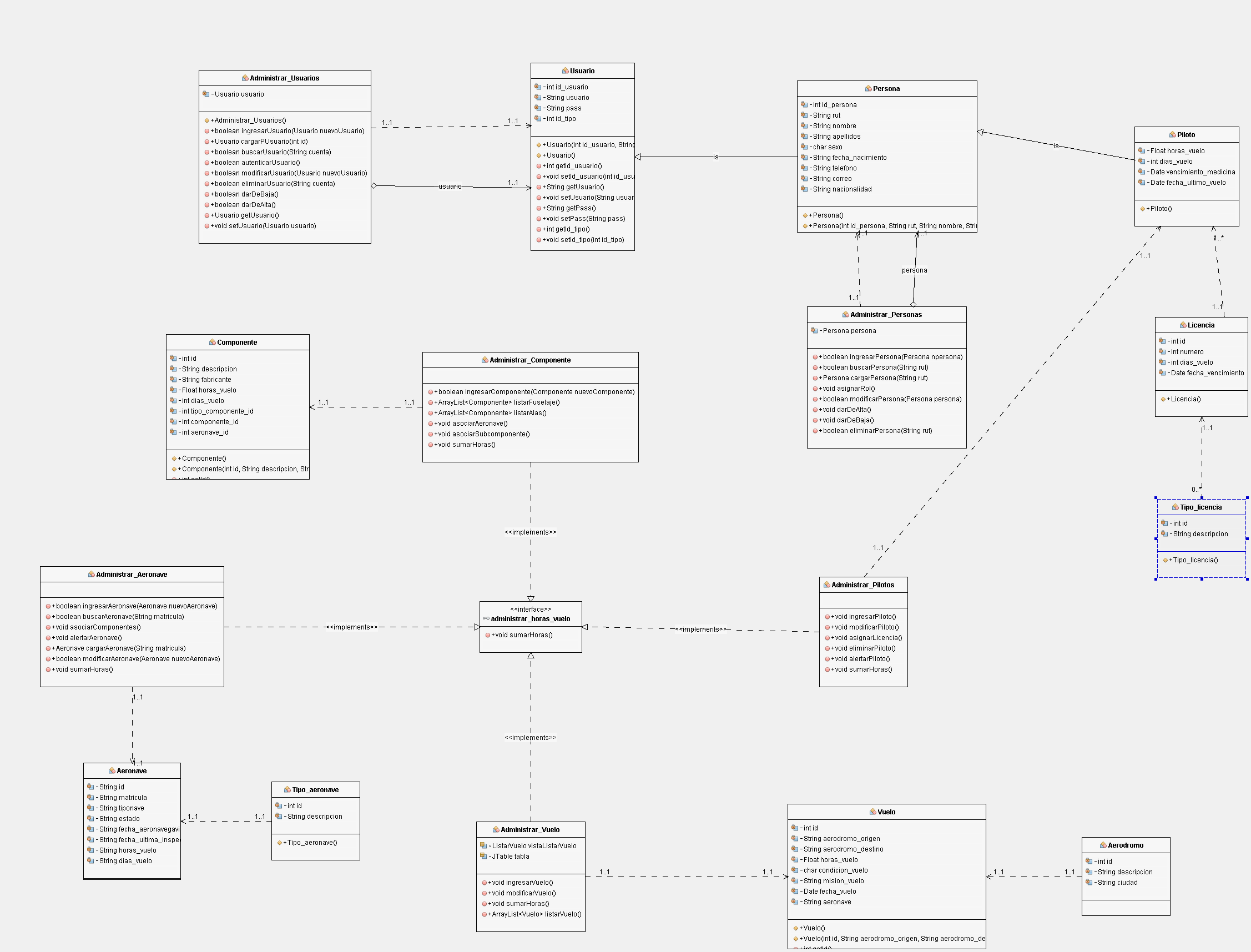


Ilustración 2 Lógica

### 3.4.3 Architectural views DESARROLLO.

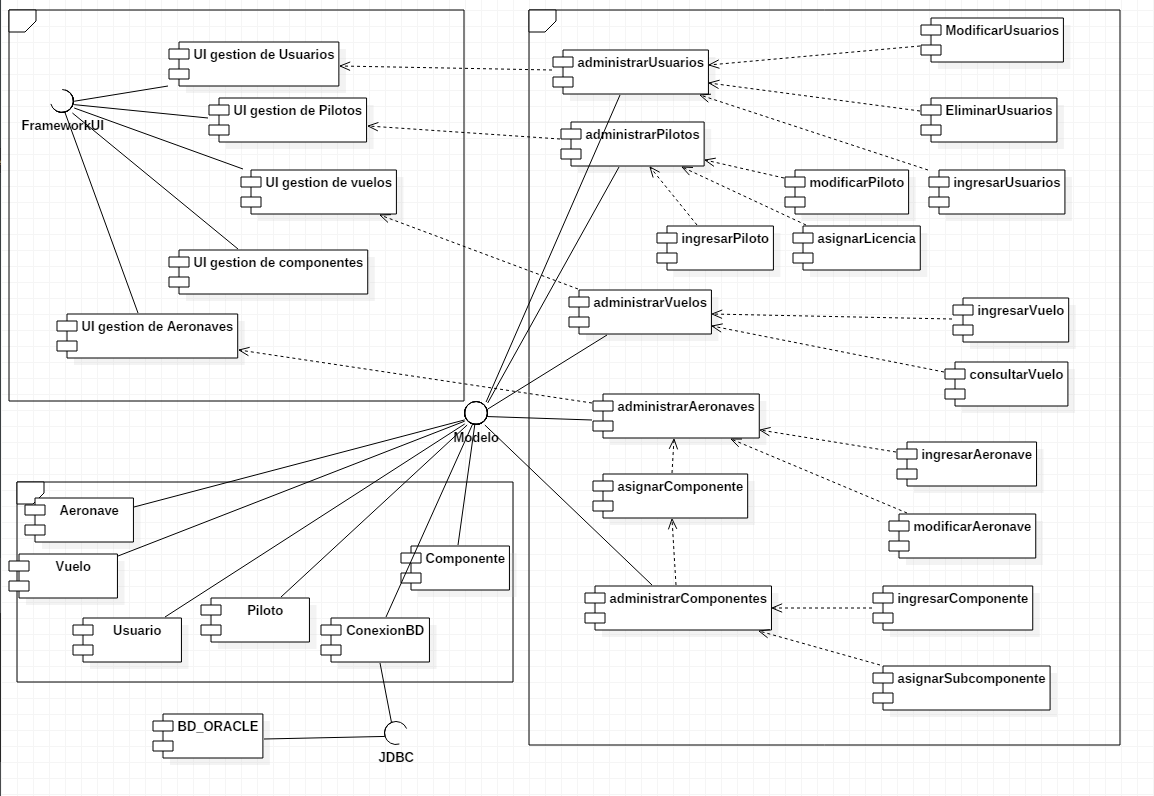
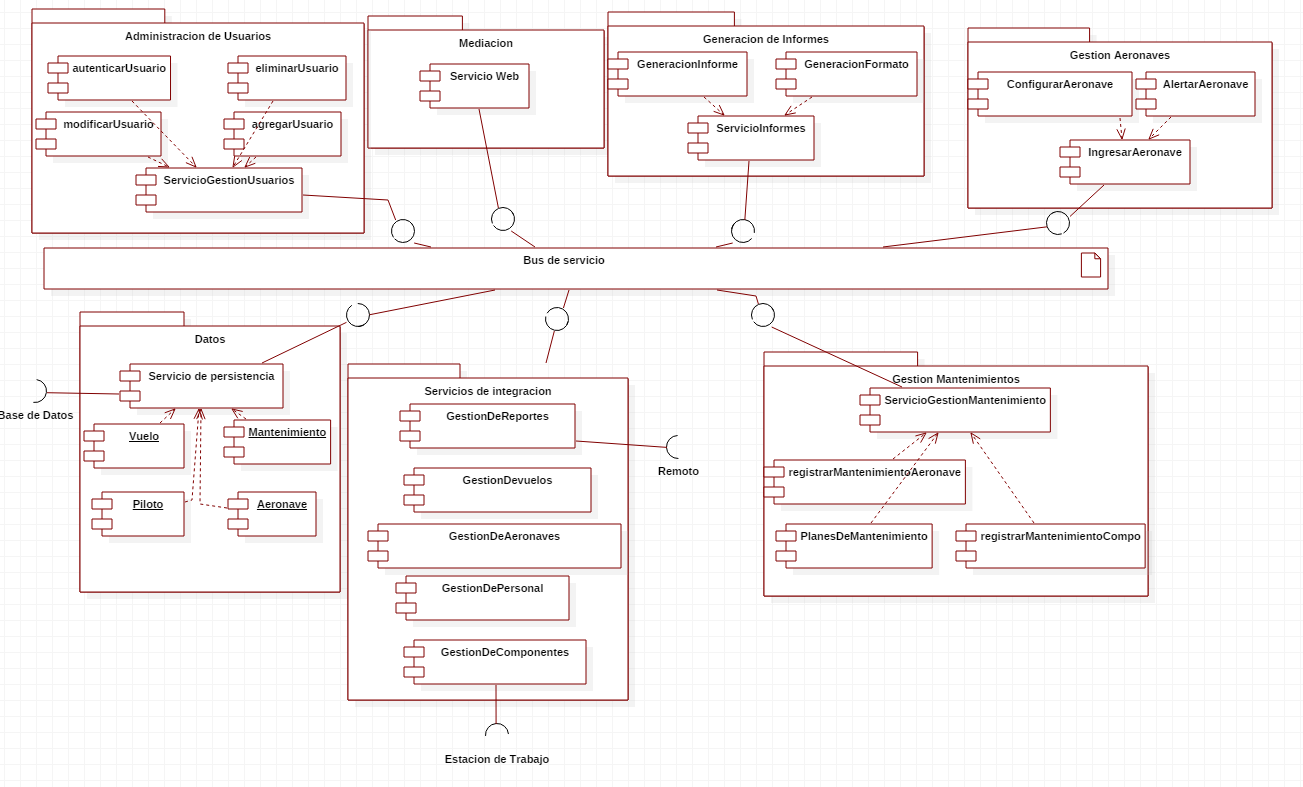


Ilustración 3 Desarrollo



### 3.4.4 Architectural views FÍSICA

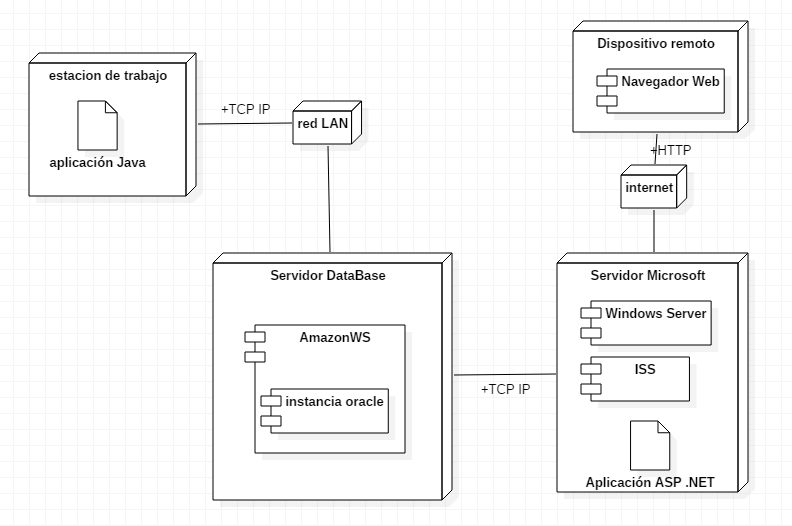


Ilustración 4 Física

### 3.4.5 Architectural views PROCESOS.

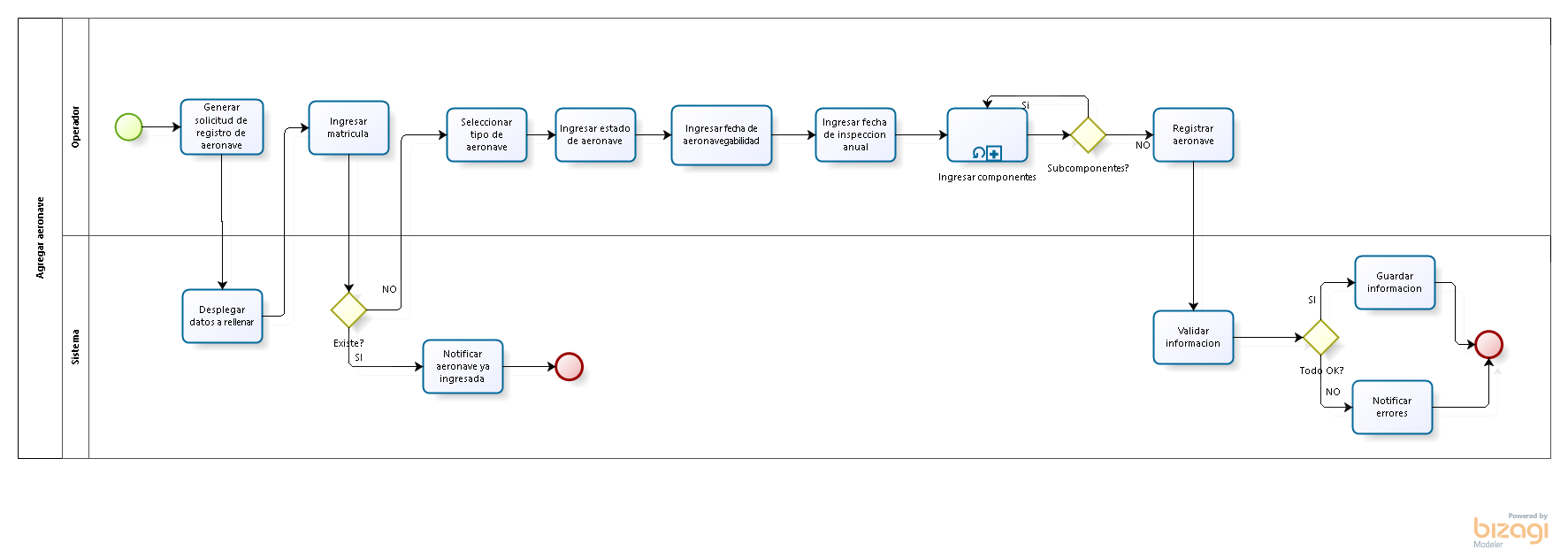
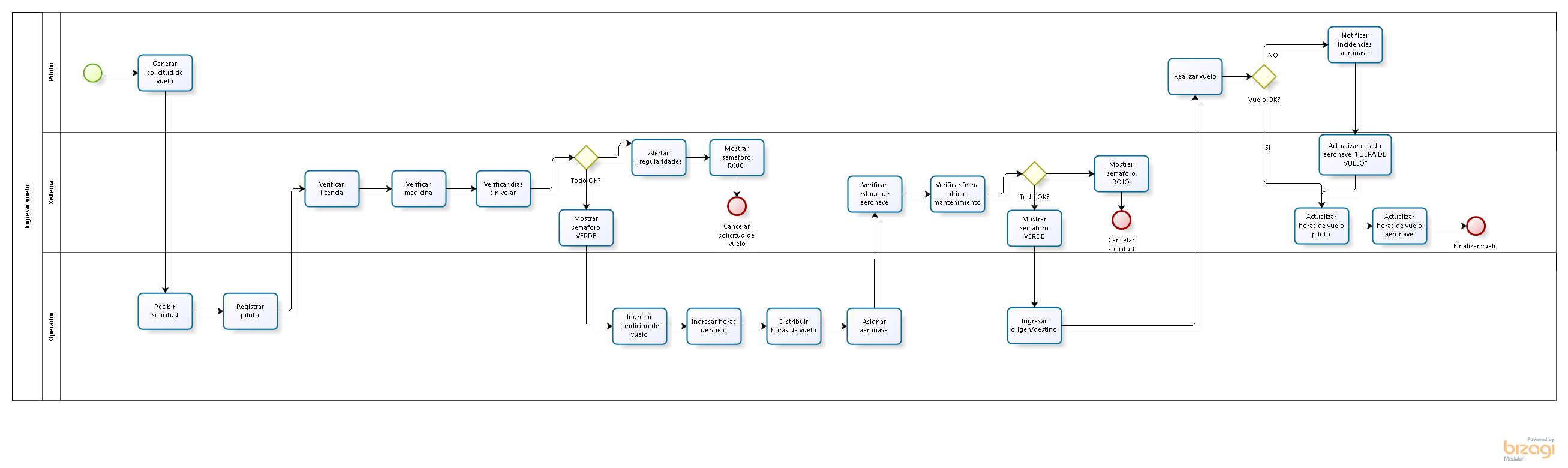
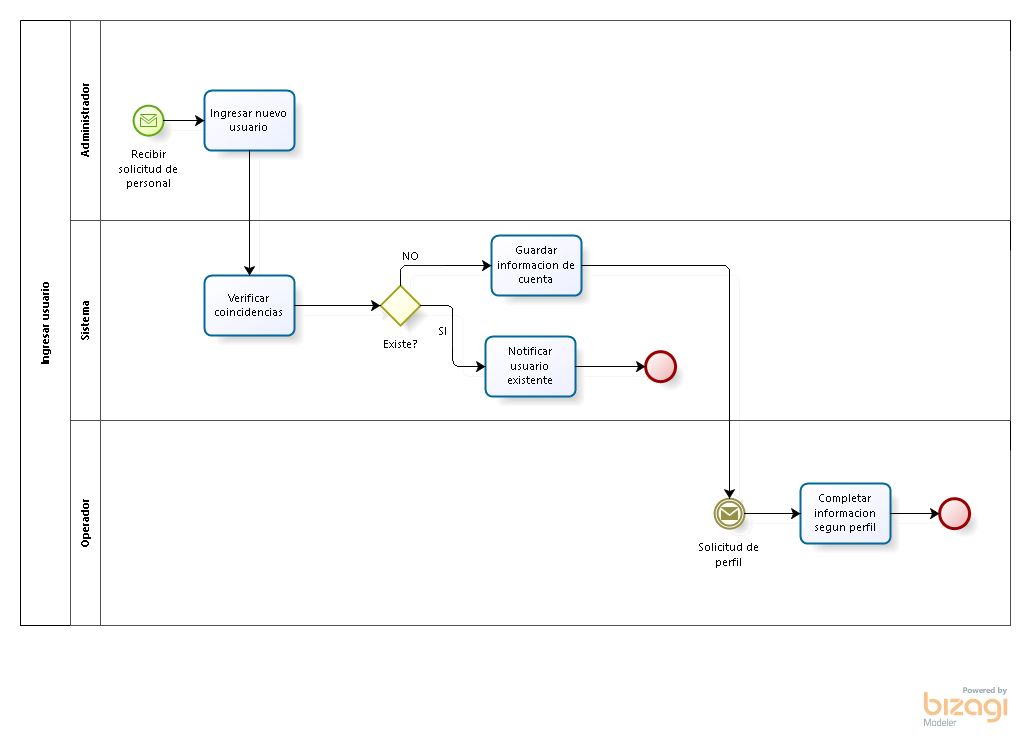


Ilustración 5 Procesos



### 3.4.6 Architectural views DATOS.

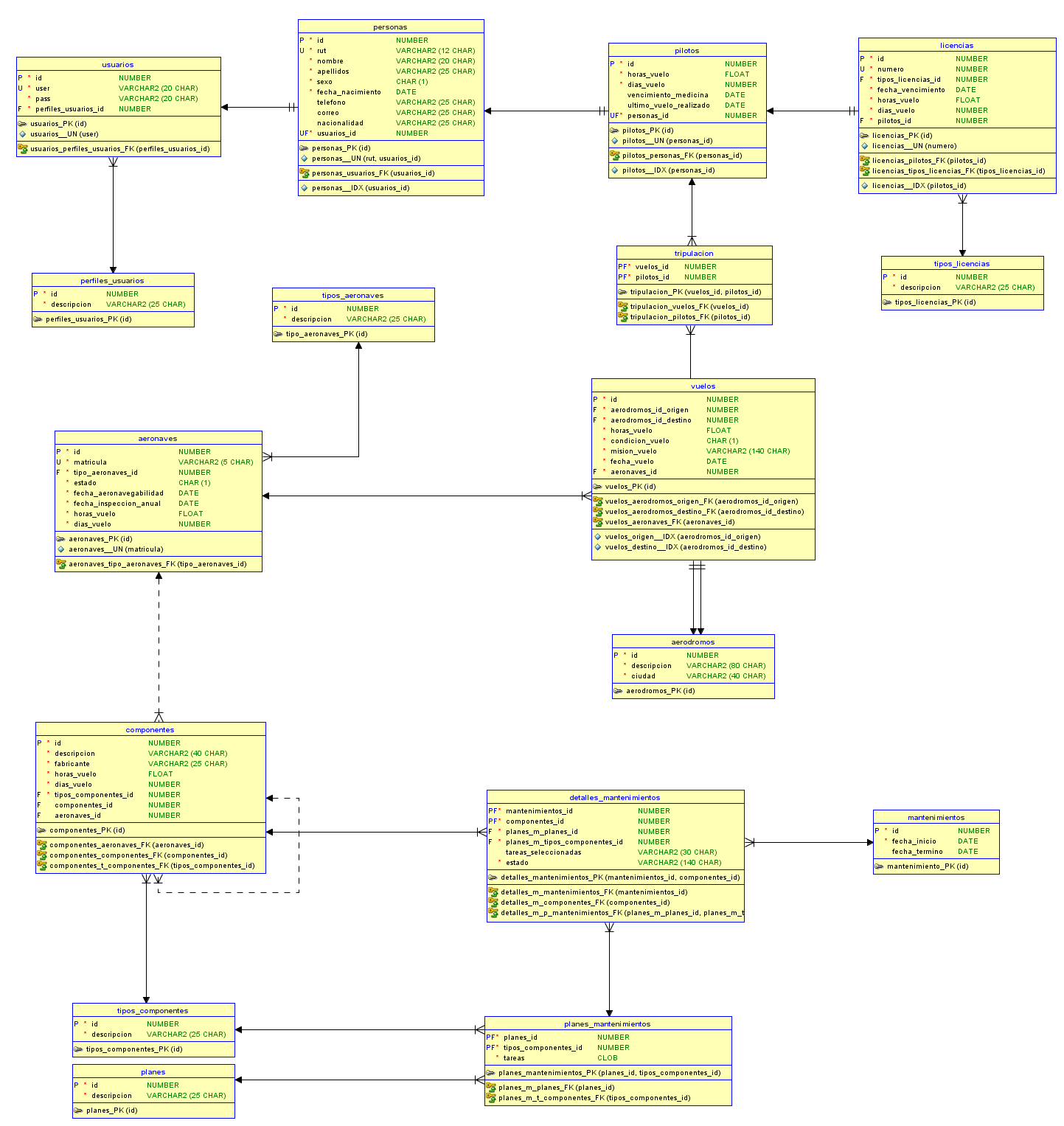


Ilustración 6 Datos

|  |
| --- |
| 3.5 Consistency among architectural views |

En esta sección se trasforma la vista de componentes a una vista arquitectural según uno o más ESTILOS ARQUITECTÓNICOS

**MVC:**

**DESCRIPCIÓN DE Módulos O PAQUETES**

Tabla 4 Modulos

|  |  |  |
| --- | --- | --- |
| **Nombre del módulo** | **descripción** | **Componentes incluidos** |
| Vista | Representa una interfaz gráfica que otorga una vista al sistema informático. | UI gestión de usuarios  UI gestión de pilotos  UI gestión de aeronaves  UI gestión de componentes  UI gestión de vuelos  FrameworkUI |
| Modelo | Módulo que contiene la definición y manejo de persistencia de datos. | aeronave, piloto, usuario, vuelo, componente, conexión a la BD. |
| Controlador | Módulo que contiene la lógica de negocio. | Administrar\_usuarios  ingresarUsuario  ModificarUsuario  EliminarUsuario  Administrar\_pilotos  ingresarpiloto  Modificarpiloto  Eliminarpiloto  Administrar\_vuelos  ingresarvuelo  Modificarvuelo  Eliminarvuelo  Administrar\_componentes  ingresarcomponente  Modificarcomponente  Eliminarcomponente |

**DESCRIPCIÓN DE COMPONENTES**

|  |  |  |
| --- | --- | --- |
| **Nombre del componente** | **descripción** | **Componentes relacionados** |
| UI gestión de usuarios | Es la interfaz gráfica donde se accede a la gestión de usuarios. | AdministrarUsuarios |
| UI gestión de pilotos | Es la interfaz gráfica donde se accede a la gestión de pilotos. | Administrarpilotos |
| UI gestión de aeronaves | Es la interfaz gráfica donde se accede a la gestión de aeronaves. | AdministrarAeronaves |
| UI gestión de componentes | Es la interfaz gráfica donde se accede a la gestión de componentes. | AdministrarComponentes |
| UI gestión de vuelos | Es la interfaz gráfica donde se accede a la gestión de vuelos. | AdministrarVuelos |
| FrameworkUI | Componente encargado de proveer de interfaz gráfica mejorada a todas las demás interfaces. | AdministrarUsuarios  Administrarpilotos  AdministrarAeronaves  AdministrarComponentes  AdministrarVuelos |
| Administrar\_usuarios | Encargado de administrar los usuarios del sistema | UIAdministrarUsuarios  Modelo |
| ingresarUsuario | Encargado del registro de nuevos usuarios | Administrar\_usuarios |
| ModificarUsuario | Encargado de la modificación de usuarios existentes | Administrar\_usuarios |
| EliminarUsuario | Encargado de la eliminación de un usuario existente | Administrar\_usuarios |
| Administrar\_pilotos | Encargado de administrar los pilotos del sistema | UI gestión de pilotos |
| ingresarpiloto | Encargado del registro de nuevos pilotos | Administrar\_pilotos |
| AsignarLicencia | Encargado de asignar la licencia a los nuevos pilotos | ingresarpiloto |
| AdministrarVuelo | Encargado de administrar los vuelos | Ingresarvuelo  consultarVuelo |
| ingresarvuelo | Encargado de ingresar los vuelos en el sistema | AdministrarVuelo |
| consultarVuelo | Encargado de consultar vuelos | AdministrarVuelo |
| AdministrarAeronave | Encargado de administrar las aeronaves | ingresarAeronave  modificarAeronave  asignarComponente |
| ingresarAeronave | Encargado de realizar el ingreso de una aeronave | AdministrarAeronave |
| modificarAeronave |  | AdministrarAeronave |
| asignarComponente | Encargado de asignar un componente a la aeronave | AdministrarAeronave |
| AdministrarComponentes | Encargado de administrar los componentes | ingresarComponente  asignarSubcomponete |
| ingresarComponente | Encargado de ingresar nuevos componentes | AdministrarComponentes |
| asignarSubcomponete | Encargado de asignar un subcomponente al componente | AdministrarComponentes |
| ConexiónBD | Encargado de realizar las transacciones con la Base de Datos | AdministrarComponentes  AdministrarAeronave  AdministrarVuelo  AdministrarPilotos  AdministrarUsuarios |
|  |  |  |

**DESCRIPCIÓN DE CONECTORES**

Tabla 5 conectores

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre conector** | **Tipo de componente** | **relaciona** | **Descripción de contenido** |
| Interface | conexión | Componentes | relaciona las diferentes capas |
| Dependencia | Dependencia | Componentes de un Módulo | indica la dependencia entre componentes. |

|  |
| --- |
| SECCION 4: REQUERIMIENTOS DE CALIDAD o ARCHITECTURAL RATIONALE |

|  |
| --- |
| 4.1 Árbol de Utilidad |

En esta sección se establecen los atributos de calidad (o requisitos no funcionales definidos en la 830) con base a los cuales ha de diseñarse el sistema. El árbol grafica los atributos de calidad según la priorización.

Se utilizaron los nombres de la sub característica – CARACTERÍSTICA de la norma de calidad 9126

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre Atributo de Calidad:**  Portabilidad | | **CARACTERÍSTICA**  Conjunto de atributos relacionados con la capacidad de un sistema software para ser transferido desde una plataforma a otra. | |
| sub característica | ID | Descripción | Prioridad |
| \*Adaptabilidad | RNF001 | El software debe adaptarse tanto a plataformas computacionales como en móviles. | Alta |

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre Atributo de Calidad:**  Mantenibilidad | | **CARACTERÍSTICA**  Conjunto de atributos relacionados con la facilidad de extender, modificar o corregir errores en un sistema software. | |
| sub característica | ID | Descripción | Prioridad |
| \*Capacidad de ser cambiado | RNF 04 | El software debe ser desarrollado de forma que éste pueda ser modificado en caso de cualquier cambio que se le necesite realizar. | Alta |

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre Atributo de**  **Calidad:**  Eficiencia | | **CARACTERÍSTICA**  Conjunto de atributos relacionados con la relación entre el nivel de desempeño del software y la cantidad de recursos necesitados bajo condiciones establecidas. | |
| sub característica | ID | Descripción | Prioridad |
| \*Comportamiento en el tiempo. | RNF 06 | El sistema debe ser capaz de proporcionar tiempos de respuesta, tiempos de proceso y potencia apropiados, bajo condiciones determinadas. | Alta |

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre Atributo de Calidad:**  Usabilidad | | **CARACTERÍSTICA**  Conjunto de atributos relacionados con el esfuerzo necesario para su uso, y en la valoración individual de tal uso, por un establecido o implicado conjunto de usuarios. | |
| sub característica | ID | Descripción | Prioridad |
| Facilidad de Aprendizaje | RNF 05 | El sistema debe ser intuitivo para el usuario, con la redacción de mensajes adecuados. | Media |

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre Atributo de Calidad:**  Funcionalidad | | **CARACTERÍSTICA**  Conjunto de atributos que se relacionan con la existencia de un conjunto de funciones y sus propiedades específicas. Las funciones son aquellas que satisfacen las necesidades implícitas o explícitas. | |
| sub característica | ID | Descripción | Prioridad |
| \*Interoperabilidad | RNF 03 | El sistema debe ser capaz de interactuar con una plataforma web desarrollada en un lenguaje de programación diferente a la de escritorio. | Media |
| \*Exactitud | RNF 02 | El sistema debe ser capaz de proporcionar resultados exactos a los procesos, cálculos y operaciones que realiza. | Media |

|  |
| --- |
| 4.2 Escenarios de Calidad Priorizados |

|  |  |  |  |
| --- | --- | --- | --- |
| **Escenario de Calidad #** | 1 |  | **Prioridad: Alta** |
| **Atributo de Calidad** | MANTENIBILIDAD – Capacidad de ser cambiado | | |
| **Justificación** | El software debe ser desarrollado de forma que éste pueda ser modificado en caso de cualquier cambio que se le necesite realizar. | | |
| **Fuente** | Interno al sistema. | | |
| **Estímulo** | Realizar una modificación o actualización al software. | | |
| **Artefacto** | Código interno del software. | | |
| **Ambiente** | Operación normal. | | |
| **Respuesta** | El software debe ser desarrollado en capas o similar. | | |
| **Medida de la Respuesta** | Medición en porcentaje de errores. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Escenario de Calidad #** | 2 |  | **Prioridad: Alta** |
| **Atributo de Calidad** | EFICIENCIA - Comportamiento en el tiempo | | |
| **Justificación** | El sistema debe ser capaz de proporcionar tiempos de respuesta, tiempos de proceso y potencia apropiados, bajo condiciones determinadas. | | |
| **Fuente** | Interno al sistema. | | |
| **Estímulo** | Envío de datos al servidor. | | |
| **Artefacto** | Software / Servidor | | |
| **Ambiente** | Operación normal. | | |
| **Respuesta** | Datos enviados correctamente al servidor. | | |
| **Medida de la Respuesta** | Menos de 4 segundos. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Escenario de Calidad #** | 3 |  | **Prioridad: Alta** |
| **Atributo de Calidad** | USABILIDAD - Facilidad de Aprendizaje | | |
| **Justificación** | El sistema debe ser intuitivo para el usuario, con la redacción de mensajes adecuados | | |
| **Fuente** | Interno al sistema. | | |
| **Estímulo** | Uso del software. | | |
| **Artefacto** | Módulos del software / Interfaz gráfica. | | |
| **Ambiente** | Operación normal. | | |
| **Respuesta** | Flujo correcto con tips de guía. | | |
| **Medida de la Respuesta** | Menor o igual a 1 segundo. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Escenario de Calidad #** | 4 |  | **Prioridad: Media** |
| **Atributo de Calidad** | FUNCIONALIDAD - Exactitud | | |
| **Justificación** | El sistema debe ser capaz de proporcionar resultados exactos a los procesos, cálculos y operaciones que realiza. | | |
| **Fuente** | Interno al sistema. | | |
| **Estímulo** | Visualizar datos del servidor. | | |
| **Artefacto** | Servidor / Software. | | |
| **Ambiente** | Operación normal. | | |
| **Respuesta** | Despliegue de información solicitada. | | |
| **Medida de la Respuesta** | Menos de 4 segundos. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Escenario de Calidad #** | 5 |  | **Prioridad: Media** |
| **Atributo de Calidad** | FUNCIONALIDAD - Interoperabilidad | | |
| **Justificación** | El sistema debe ser capaz de interactuar con una plataforma web desarrollada en un lenguaje de programación diferente a la de escritorio. | | |
| **Fuente** | Interno al sistema. | | |
| **Estímulo** | Visualizar datos del servidor en plataforma web y escritorio. | | |
| **Artefacto** | Servidor / Software. | | |
| **Ambiente** | Operación normal. | | |
| **Respuesta** | Despliegue de información solicitada. | | |
| **Medida de la Respuesta** | Menos de 4 segundos. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Escenario de Calidad #** | 6 |  | **Prioridad: Alta** |
| **Atributo de Calidad** | PORTABILIDAD - Adaptabilidad | | |
| **Justificación** | El software debe adaptarse tanto a plataformas computacionales como en móviles. | | |
| **Fuente** | Interno al sistema. | | |
| **Estímulo** | Mudanza del software de un sistema operativo a otro. | | |
| **Artefacto** | Computadores de la escuela y smartphones. | | |
| **Ambiente** | Operación normal. | | |
| **Respuesta** | El software se ejecuta correctamente sin presentar problemas durante su uso. | | |
| **Medida de la Respuesta** | La ejecución de éste no debe tardar más de 3 segundos. | | |

|  |
| --- |
| SECCION 5: Principios heurísticos de Jacob Nielsen |

Principios heurísticos de Jacob Nielsen

## 1 Control y libertad del usuario

**Descripción:**

Los usuarios frecuentemente eligen opciones por error, por eso indica una salida clara a esas situaciones no deseadas sin necesidad de pasar por extensos diálogos.

**Aplicado al proyecto:**

El software permite la eliminación de los registros insertados.

## 2 Evitar errores

**Descripción:**

Un diseño cuidado que previene problemas es mejor que unos buenos mensajes de error.

**Aplicado al proyecto:**

El sistema cuenta con activación de campos, esto quiere decir que no se mostrara la opción o no permitirá el ingreso a la opción que ejecuta código si no están validados o ingresados ciertos datos.

## 3 Diseño minimalista

**Descripción:**

No hay que mostrar información irrelevante. Cada pedazo de información extra compite con la importante y disminuye su relativa visibilidad.

**Aplicado al proyecto:**

El sistema solo muestra la información relevante para no disminuir la visibilidad del usuario.

## 4 Reconocer, diagnosticar y recuperarse de los errores

**Descripción:**

Para ayudar a los usuarios, los mensajes de error deben estar escritos en lenguaje sencillo, indicar el problema de forma precisa e indicar una solución.

**Aplicado al proyecto:**

Cada error que muestre el sistema se describe adecuadamente indicando donde ocurrió y como solucionarlo.

## 5 Correspondencia entre el sistema y el mundo real

**Descripción:**

El sistema debe [hablar el lenguaje de los usuarios](http://albertolacalle.com/contenidos.htm) con palabras, frases y conceptos familiares. Sigue las convenciones del mundo real. Haz que la información aparezca en forma natural y lógica.

**Aplicado al proyecto:**

El sistema maneja lenguaje general y especifico a la escuela de aviación.

## 

|  |
| --- |
| SECCION 6: SIMBOLOGÍA |

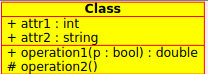
(Esta sección desarrolle un esquema de los símbolos de UML que ha usado y describa brevemente para qué sirve cada uno.)

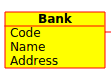
Actor: Un actor "especifica un rol jugado por un usuario o cualquier otro sistema que interactúa con el sujeto.

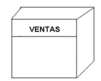
**Caso de uso**: Es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

**Asociación**: Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple.

**Dependencia o Instanciación**: Es una forma muy particular de relación entre clases, en la cual una clase depende de otra, es decir, se instancia (se crea). Dicha relación se denota con una flecha punteada.

Clase: Una clase define los atributos y los métodos de una serie de objetos. Todos los objetos de esta clase (instancias de esa clase) tienen el mismo comportamiento y el mismo conjunto de atributos (cada objeto tiene el suyo propio). En ocasiones se utiliza el término «tipo» en lugar de clase, pero recuerde que no son lo mismo, y que el término tipo tiene un significado más general.

Entidad: Una Entidad es cualquier concepto del mundo real con una existencia independiente. Puede ser un objeto con una existencia física o puede ser un objeto con una existencia conceptual. Cada entidad tiene un conjunto de atributos que describen las propiedades de la entidad.

Nodo: Es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, que generalmente tiene alguna memoria y capacidad de procesamiento.

# SECCION 6: SIMBOLOGÍA

|  |
| --- |
| SECCION 7: Framework |

## Framework Swing

Para la realización de la aplicación se usará el framework de Swing que nos permite realizar la aplicación mediante una interfaz gráfica. Este framework está definido como una infraestructura en común en la mayoría de las aplicaciones de escritorio.

Además, se incorporarán diferentes librerías para mejorar y cambiar un poco el framework tales como:

* Jtatto que se compone de diferentes Look and Feels (se refiere al aspecto de la aplicación) para las aplicaciones de Swing. En este caso se eligió el tema de Acryl de esta librería que nos permite mejorar la apariencia de la aplicación.
* Jcalendar agrega componente de calendario al framework de Swing