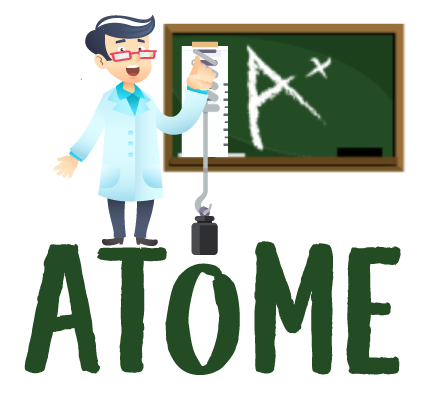
**Documento de Arquitectura de Software**

ATOME



Armenia, 17-Abr-2018

HISTORIAL DE CAMBIOS

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Fecha** | **Descripción** | **Elaborado por:** |
| 0.0.1 | 16/04/18 | Creación del documento | Yenifer Hernandez – Mauricio Uribe |

TABLA DE CONTENIDO

[2 LISTA DE FIGURAS 5](#_Toc511704069)

[3 LISTA DE TABLAS 5](#_Toc511704070)

[1. INTRODUCCIÓN 6](#_Toc511704071)

[1.1. Propósito 6](#_Toc511704072)

[1.2. Alcance 6](#_Toc511704073)

[1.3. Definición, acrónimos y abreviaturas. 7](#_Toc511704074)

[2. ARQUITECTURA DE SOFTWARE 9](#_Toc511704075)

[3. METAS Y RESTRICCIONES ARQUITECTÓNICAS 11](#_Toc511704076)

[3.1.1 Priorización de requerimientos 11](#_Toc511704077)

[3.1.2 Resumen de los requerimientos 11](#_Toc511704078)

[3.1.3 Metas 11](#_Toc511704079)

[3.1.4 Restricciones 12](#_Toc511704080)

[4. VISTAS 12](#_Toc511704081)

[4.1. VISTA DE CASOS DE USO 12](#_Toc511704082)

[4.2. VISTA LÓGICA 13](#_Toc511704083)

[4.3. VISTA DE PROCESOS 13](#_Toc511704084)

[4.4. VISTA DE IMPLEMENTACIÓN 13](#_Toc511704085)

[4.5. VISTA FISICA 14](#_Toc511704086)

[4.6. VISTA DE DATOS 14](#_Toc511704087)

[5. ESCENARIOS Y TACTICAS 14](#_Toc511704088)

[5.1.1 Rendimiento o performance 14](#_Toc511704091)

[5.1.2 Gestión de recursos 15](#_Toc511704092)

[5.1.3 Tiempo de bloqueo 15](#_Toc511704093)

[5.1.4 Seguridad 16](#_Toc511704094)

[5.1.5 Resistir ataques 16](#_Toc511704095)

[5.1.6 Modificabilidad 16](#_Toc511704096)

[5.1.7 Localización de cambios 16](#_Toc511704097)

[5.1.8 Usabilidad 17](#_Toc511704098)

[5.1.9 Separar la interfaz del resto del sistema 17](#_Toc511704099)

[5.1.10 El usuario toma la iniciativa 17](#_Toc511704100)

[5.1.11 Diálogos simples y naturales 18](#_Toc511704101)

[5.1.12 Hablar el lenguaje del usuario 18](#_Toc511704102)

[5.1.13 Proveer salidas marcadas claramente 19](#_Toc511704103)

[5.1.14 Dar buenos mensajes de error 19](#_Toc511704104)

[5.1.15 Prevenir y manejar errores 20](#_Toc511704105)

# LISTA DE TABLAS

[Tabla 1 Definiciones 7](#_Toc511704135)

[Tabla 4 Rendimiento y performance – Gestión de Recursos 15](#_Toc511704136)

[Tabla 5 Seguridad – Resistir ataques 16](#_Toc511704137)

[Tabla 6 Modificabilidad – Localización de cambios 16](#_Toc511704138)

[Tabla 7 Usabilidad - Separar la interfaz del resto del sistema 17](#_Toc511704139)

[Tabla 8 Usabilidad - El usuario toma la iniciativa 17](#_Toc511704140)

[Tabla 9 Usabilidad – Diálogos simples y naturales 18](#_Toc511704141)

[Tabla 10 Usabilidad – Hablar el lenguaje del usuario 18](#_Toc511704142)

[Tabla 11 Usabilidad – Proveer salidas marcadas claramente 19](#_Toc511704143)

[Tabla 12 Usabilidad – Dar buenos mensajes de error 19](#_Toc511704144)

[Tabla 13 Usabilidad – Dar buenos mensajes de error 20](#_Toc511704145)

# INTRODUCCIÓN

En el presente documento se muestra una descripción detallada del diseño de alto nivel de la aplicación para la aplicación que permita automatizar la toma de las mediciones en prácticas de laboratorio del programa de física de la Universidad del Quindío (ATOME). La cual es una aplicación que busca minimiza el error humano en la toma de medidas y por consiguiente se logra más precisión en los resultados de la práctica.

Este diseño será representado por medio del modelo de “4+1 Vistas” [1] aunque se detallan otras vistas adicionales que permiten especificar el detalle de la estructura, planos y/o especificaciones del proyecto, todo con el fin de mostrar, en cada una de las vistas, una perspectiva o visión de un conjunto de elementos del proyecto y sus relaciones, esto desde el punto de vista de la arquitectura. De esta manera presentar la documentación respectiva de este proyecto de forma ordenada, clara y coherente.

## Propósito

El propósito de este documento es especificar claramente la arquitectura de software a ser usada para el desarrollo de la ATOME. Con este documento se busca plasmar en términos arquitectónicos y de diseño todos los requerimientos definidos en el documento de visión y de requisitos generado al inicio del proyecto con el cliente en días anteriores.

Este documento va dirigido a los stakeholders implicados en el proceso de desarrollo del software. Estos son para este proyecto los siguientes en mención:

* Director del programa de Física de la Universidad del Quindío - Product Owner.
* Estudiantes de la Universidad del Quindío que tengan cursos adscritos y/o dirigidos por el Programa de Física de la Universidad del Quindío.
* Personal administrativo de la aplicación y/o técnicos y auxiliares de los laboratorios adscritos al programa de Física de la Universidad del Quindío.
* Todos los demás interesados en el proyecto.

Este documento es la principal fuente de información para empezar con el desarrollo del sistema y la integración de los componentes adquiridos.

## Alcance

Con este documento se pretende definir la arquitectura de software que será utilizada para el desarrollo de la aplicación ATOME. Como se verá posteriormente en este documento (ver sección Representación Arquitectónica) esta representación arquitectónica es especificada por las diferentes vistas del sistema (Casos de Uso, Lógica, Procesos, Despliegue e Implementación).También son detallados y direccionados otros aspectos importantes con respecto a los requerimientos no funcionales como lo son el tamaño, desempeño y la calidad de la arquitectura del sistema. Es preciso aclarar que estos diseños se realizan teniendo en cuenta los componentes de la plataforma Firebase para la gestión lógica del desarrollo.

## Definición, acrónimos y abreviaturas.

A continuación se lista algunas de las palabras más mencionadas durante el desarrollo y que pueden llegar a ser desconocidas o ambiguas por su definición técnica; aunque también en el pie de cada página estará la definición de las palabras con esta característica.

Tabla Definiciones

|  |  |
| --- | --- |
| Palabra o Concepto | Definición |
| Ionic | Ionic es un framework gratuito y open source para desarrollar aplicaciones híbridas multiplataforma que utiliza HTML5, CSS (generado por SASS) y Cordova como base. Es uno de los framework del momento por utilizar AngularJS para gestionar las aplicaciones, lo que asegura aplicaciones rápidas y escalables. |
| Firebase | Servicio capaz de proveernos de un Backend en la nube con una fuente de datos en tiempo real y librerías para acceder a ella desde aplicaciones web, iOS y Android. |
| node.js | es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Node.js usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente. El ecosistema de paquetes de Node.js, npm, es el ecosistema más grande de librerías de código abierto en el mundo. |
| cordova | es un framework para el desarrollo de aplicaciones móviles propiedad de Adobe Systems que permite a los programadores desarrollar aplicaciones para dispositivos móviles utilizando herramientas web genéricas como JavaScript, HTML5 y CSS3, resultando aplicaciones híbridas. |
| angularJs | Es un proyecto de código abierto, realizado en Javascript que contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web y propone una serie de patrones de diseño para llevarlas a cabo. En pocas palabras, es lo que se conoce como un framework para el desarrollo, en esta caso sobre el lenguaje Javascript con programación del lado del cliente. |
| typescript | es un lenguaje de programación de código abierto desarrollado y presentado por Microsoft hace unos tres años. Es un superconjunto de JavaScript que esencialmente añade capacidades de POO como es el tipado estático y objetos basados en clases. |
| servicio auth-jwt | es un método de autenticación mediante tokens |
| cloudMessagin | es un servicio gratuito que nos permite enviar información desde nuestro servidor a un dispositivo Android. El servicio GCM controla todo lo relacionado con el almacenamiento en cola de los mensajes y su entrega a las aplicaciones. |
| No-SQL | es un término que describe las bases de datos no relacionales de alto desempeño. Las bases de datos NO SQL utilizan varios modelos de datos, incluidos los de documentos, gráficos, claves-valores y columnas. Las bases de datos NO SQL son famosas por la facilidad de desarrollo, el desempeño escalable, la alta disponibilidad y la resiliencia. A continuación, presentamos varios recursos que le servirán para comenzar a usar las bases de datos NO SQL. |
| Casos de Uso | Describe el conjunto de escenarios y casos de uso que representan alguna funcionalidad central del sistema. |
| Vista Lógica | Ofrece soporte a los requerimientos funcionales, lo que el sistema debe proveer en términos de servicios a sus usuarios. En el presente documento, en la vista lógica se mostrará la división del sistema en paquetes. |
| Vista de Procesos | La vista de procesos permite describir los procesos del sistema y como estos se comunican. |
| Vista Física | En esta vista se muestra desde la perspectiva de un ingeniero de sistemas todos los componentes físicos del sistema así como las conexiones físicas entre esos componentes que conforman la solución (incluyendo los servicios). |
| Vista de desarrollo o de implementación | Esta vista sirve para unir a las otras cuatro por medio de secuencias de interacciones que se llegan a establecer entre sus elementos. Tiene dos propósitos, sirve como una guía arquitectónica para descubrir elementos arquitectónicos en el momento del diseño, y para validar e ilustrar el diseño de la arquitectura. |

# ARQUITECTURA DE SOFTWARE

Con el fin de describir la arquitectura de la aplicación para el centro comercial MegaCity de acuerdo a los requerimientos por pate del cliente, el tiempo y equipo de trabajo, se analizan diferente estructuras y tecnologías; después de indagar e investigar se toma la decisión de trabajar con diferentes arquitecturas, como cliente servidor para la lógica del negocio y mvvm para el desarrollo de la aplicación. Lo anteriormente descrito se encuentra graficado en la siguiente imagen, donde claramente se evidencian las tecnologías que se están implementando y la manera como estas interactúa.

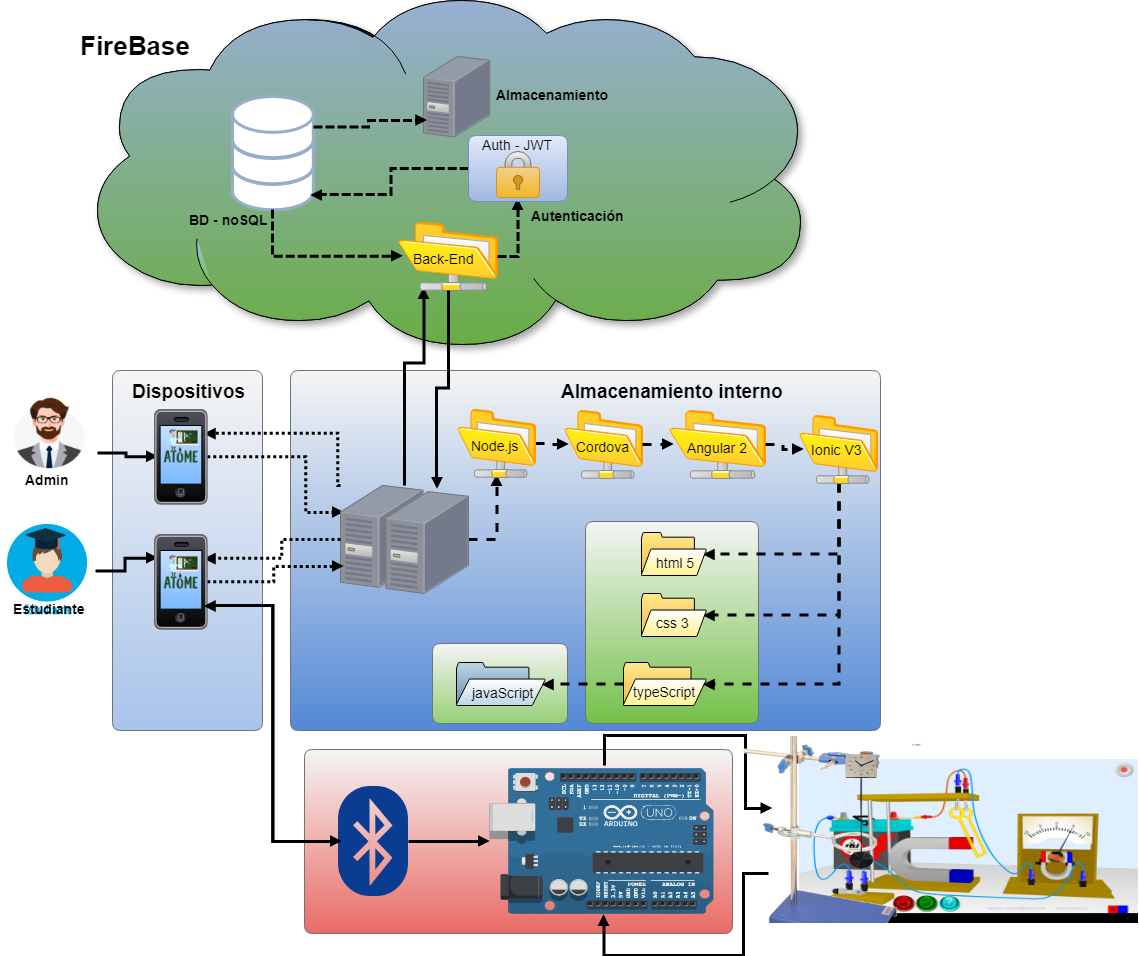


Figura arquitectura ATOME

De acuerdo con la figura anterior se nota el uso del framework Ionic[[1]](#footnote-1) en su versión para el front-end[[2]](#footnote-2) o construcción de las pantallas o entorno visual de la aplicación; para la gestión de la lógica del negocio, back-end[[3]](#footnote-3) se utiliza la plataforma Firebase[[4]](#footnote-4).

Entonces a continuación se describe las dos divisiones de la arquitectura de la aplicación:

* **Aplicación Móvil:**
  1. **Back-End**: este es implementado sobre la plataforma Firebase de google, en la nube, gratuito, la cual provee una gestión del código del proyecto, como un almacenamiento en tiempo real sobre una base de datos No SQL[[5]](#footnote-5), además unos servicios adicionales y de vital importancia para la ejecución de este proyecto, como la autenticación JWT[[6]](#footnote-6), el cloud messagin[[7]](#footnote-7) , un hosting para el despliegue de la página web del sitio y un almacenamiento directo en el servidor (storage).
  2. **Front-End**: se realiza haciendo uso del framework Ionic, el cual se compone de un gestor de servicios node.js v6, framework de aplicaciones Cordova[[8]](#footnote-8), librerías de AngularJs[[9]](#footnote-9), permitiendo realizar entornos visuales de manera ágil haciendo uso de herramientas emergentes como html5, css3, y typescript[[10]](#footnote-10) (un conjunto empaquetado de JavaScript).
* **Aplicación embebida:**

1. **Programación del Arduino:** Se utiliza Arduino Mega, en el cual se encuentra el control de la conexión bluetooh con el dispositivo móvil, además del control de los sensores que se encuentran en estructura física, mediante librerías y subrutinas.
2. **Adecuación de estructura física con sensores:** Montaje físico de la práctica de laboratorio, con los sensores necesarios para la toma de variables que permiten la captura de los eventos generados durante la práctica de laboratorio.

Adicional a esta descripción se emplea el modelo de vistas múltiples con el fin de describir aún más la aplicación ATOME; este modelo organiza una descripción de la arquitectura del software utilizando cinco vistas concurrentes, permitiendo aproximar de manera aislada los intereses de los stakeholders de la arquitectura: los usuarios finales, el equipo de desarrollo, entre otros; y manejar de manera separada los requerimientos funcionales[[11]](#footnote-11).

Entonces empleamos los siguientes diagramas: De casos de uso, de paquetes, de despliegue, de estados, de secuencia y de clases, adicionalmente se definió un modelo de datos, todos estos diagramas serán descritos más adelante dentro de este documento.

# METAS Y RESTRICCIONES ARQUITECTÓNICAS

### Priorización de requerimientos

De acuerdo a lo dialogado con el cliente, se realizó la distribución de todas las funcionalidades o experiencias de usuario en dos fases de dos sprint’s o entregables cada una, en donde definimos fechas de entrega lo cual quedo diligenciado y acordado con el cliente en el documento de iniciación del proyecto.

### Resumen de los requerimientos

En la siguiente tabla se enumeran, describen y valoran cada uno de los requerimientos, que para facilitar el ciclo de vida del proyecto se proyectaron como historias de usuario.

1. **Fase 1:**

El alcance de esta fase es definida de acuerdo al proyecto del curso de Software 3.

* Sprint 1:

(poner los requisitos)

* Sprint 2:

(poner los requisitos)

1. **Fase 2:**

Los demás requisitos solicitados por el cliente, comprendido como proyecto de grado de los desarrolladores de la misma.

* Sprint 1:

(poner los requisitos)

* Sprint 2:

(poner los requisitos)

### Metas

Una vez establecido el objetivo central de la aplicación o lo que el cliente desea, que es minimizar el error humano en la toma de medidas y por consiguiente se logra más precisión en los resultados de la práctica. Se disponen las siguientes metas:

* La aplicación debe autenticar el estudiante.
* La aplicación debe permitir seleccionar estaciones de trabajo.
* La aplicación debe permitirle iniciar la práctica de laboratorio de acuerdo a la estación de trabajo seleccionada.
* La aplicación debe indicarle al estudiante el valor de las variables obtenidas durante la ejecución de la práctica.
* La aplicación le debe permitir al estudiante ejecutar n veces la práctica hasta que su bitácora de trabajo no este un estado terminada.
* La aplicación debe generar un listado de medidas obtenidas durante la ejecución de la práctica de laboratorio.
* La aplicación debe generar una gráfica con las medidas realizadas.
* La aplicación le debe permitir al cliente ver las bitácoras creadas por él.
* La aplicación debe permitir agregar miembros de equipo a la bitácora.
* La aplicación le debe permitir compartir el pdf generado a los miembros del equipo utilizando el correo electrónico.
* La aplicación le debe permitir al administrador crear nuevas estaciones de trabajo.

### Restricciones

Un vez establecido que se va a realizar dentro de la aplicación se listan las siguientes restricciones:

* El acceso para los estudiantes debe ser haciendo uso del correo institucional.
* Para iniciar una práctica, el dispositivo bluetooh del teléfono debe estar conectado correctamente a la estación de trabajo.

Pendiente definir mas

# VISTAS

## VISTA DE CASOS DE USO

## VISTA LÓGICA

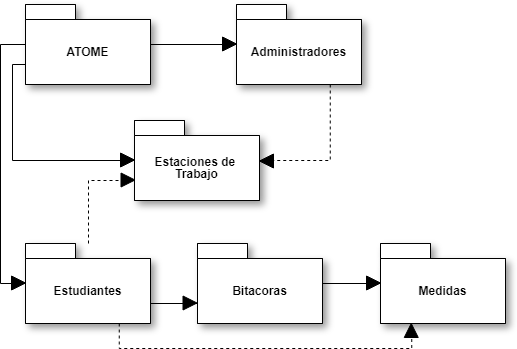


Figura Diagrama de Paquetes

Figura Diagrama de Secuencia

## VISTA DE PROCESOS

Figura Diagrama de Clases

## VISTA DE IMPLEMENTACIÓN

Figura Diagrama de Estados

## VISTA FISICA

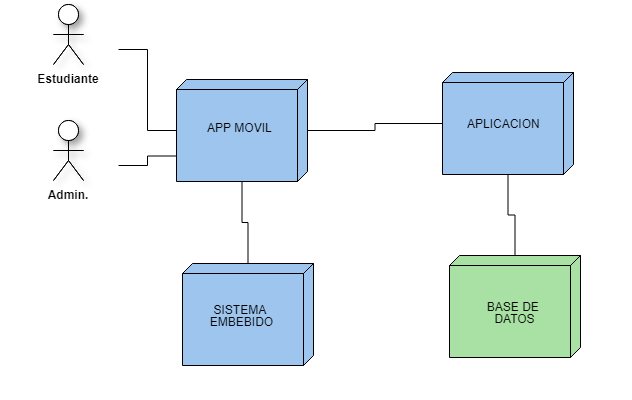


Figura Diagrama de Despliegue

## VISTA DE DATOS

En esta vista se explica la manera como fueron almacenados los datos en la base de datos de la aplicación ATOME. Esta vista es vital porque el sistema se encarga de gestionar estaciones de trabajo, bitácoras y usuarios.

Figura Modelo de Datos

# ESCENARIOS Y TACTICAS

En esta sección se describen las características de software que afectan de manera directa la arquitectura de la aplicación para el centro comercial ATOME, como los son las tácticas que aportan atributos de calidad y de esta manera medir y evaluar la satisfacción del usuario.



### Rendimiento o performance

Habilidad de un sistema para responder a un evento que ocurre (interrupciones, mensajes, pedidos de usuarios o paso del tiempo) dentro de un perfil definido.

### Gestión de recursos

Tabla Rendimiento y performance – Gestión de Recursos

|  |  |
| --- | --- |
|  | |
| **Atributo de Calidad** | Rendimiento y performance – Gestión de Recursos |
| **Justificación** | Aprovechando la ventaja de gestión de código que ofrece la plataforma Firebase de Google, en la cual se permite configurar hasta 100 usuarios concurrentes, para lo cual se entiende que dentro de la aplicación ATOME pueden 100 usuarios consultar al mismo tiempo los datos. |
| **Fuente** | Sistema |
| **Estimulo** | Concurrencia de usuarios ejecutando transacciones. |
| **Artefacto** | Sistema |
| **Ambiente** | Producción |
| **Respuesta** | Distribución correcta de recursos entre usuarios conectados |

### Tiempo de bloqueo

Se refiere a cuando una computación al usar un recurso se detiene momentáneamente debido a: Contención del recurso: Ocurre cuando varias peticiones llegan al mismo momento. Entre más peticiones más latencia; Disponibilidad del recurso: Cuando el recurso no está disponible por alguna falla momentánea; Dependencia hacia otra computación: Cuando por ejemplo debe esperar los resultados de otro proceso para computarlos en conjunto y dar así una respuesta.

Será más alta la latencia si los procesos están en serie y no en paralelo.

|  |  |
| --- | --- |
|  | |
| **Atributo de Calidad** | Rendimiento y performance – Tiempo de bloqueo |
| **Justificación** | La aplicación ATOME cuenta con notificaciones de estado de disponibilidad de los dispositivos hardware implementados. |
| **Fuente** | Sistema |
| **Estimulo** | Selección de estaciones de trabajo |
| **Artefacto** | Sistema |
| **Ambiente** | Producción |
| **Respuesta** | Selección de dispositivos hardware libres y disponibles para su conexión |

### Seguridad

Habilidad de un sistema para controlar, monitorear y auditar en forma confiable quién puede realizar qué acciones sobre el sistema y sus recursos, y la habilidad para detectar y recuperarse de fallas en los sistemas de seguridad.

### Resistir ataques

Tabla Seguridad – Resistir ataques

|  |  |
| --- | --- |
|  | |
| **Atributo de Calidad** | Seguridad – Resistir ataques |
| **Justificación** | La aplicación ATOME cuenta con un servicio de autenticación por tokens, lo cual una vez que el usuario se autentica, el sistema le asigna un identificador único para la sesión activa del usuario, una vez cierra su sesión este acceso o este código único caduca, por ende nadie puede clonar la sesión y acceder de manera incorrecta o indebida |
| **Fuente** | Sistema |
| **Estimulo** | Ataques a bases de datos |
| **Artefacto** | Información |
| **Ambiente** | Producción |
| **Respuesta** | Validación de información y sesiones de ingreso |

### Modificabilidad

La modificabilidad de un sistema se refiere al atributo de calidad de una arquitectura software relativo al coste del cambio y a la facilidad con que un sistema informático puede adaptarse a los cambios.

### Localización de cambios

Tabla Modificabilidad – Localización de cambios

|  |  |
| --- | --- |
|  | |
| **Atributo de Calidad** | Modificabilidad – Localización de cambios |
| **Justificación** | La aplicación ATOME cuenta una división del código y la interfaz gráfica, además una división de responsabilidades por función, facilitando la comprensión del mismo y la ubicación de cada una dentro de la estructura del proyecto. |
| **Fuente** | Usuarios (Administradores) |
| **Estimulo** | Forma en que interactúa el desarrollador (administradores) con el software. |
| **Artefacto** | Sistema |
| **Ambiente** | Desarrollo |

### Usabilidad

La facilidad con la cual las personas interactúan con la aplicación en forma efectiva, y sobre como el sistema provee soporte al usuario en este sentido.

### Separar la interfaz del resto del sistema

Tabla Usabilidad - Separar la interfaz del resto del sistema

|  |  |
| --- | --- |
|  | |
| **Atributo de Calidad** | Usabilidad - Separar la interfaz del resto del sistema |
| **Justificación** | La aplicación ATOME cuenta una división del código y la interfaz gráfica, tanto a nivel estructural como de responsable dentro del equipo de trabajo. |
| **Fuente** | Usuarios (Administradores) |
| **Estimulo** | Forma en que interactúa el desarrollador (administradores) con el software. |
| **Artefacto** | Sistema |
| **Ambiente** | Desarrollo |
| **Respuesta** | El administrador deberá poder acceder al código responsable de la interfaz gráfica sin afectar el código responsable de la lógica y los datos de la aplicación. |

### El usuario toma la iniciativa

Tabla Usabilidad - El usuario toma la iniciativa

|  |  |
| --- | --- |
|  | |
| **Atributo de Calidad** | Usabilidad - El usuario toma la iniciativa |
| **Justificación** | La aplicación ATOME cuenta con botones visibles como de cancelar, como confirmaciones antes de ejecutar acciones transaccionales, otorgándole así la responsabilidad al usuario que se encuentra manipulando los datos. |
| **Fuente** | Usuarios (Estudiantes y Administradores) |
| **Estimulo** | Forma en que interactúa el usuario final sobre el software final. |
| **Artefacto** | Sistema |
| **Ambiente** | Producción |
| **Respuesta** | El usuario deberá poder navegar sin problemas en el sistema y sin la ayuda de algún manual |

### Diálogos simples y naturales

Tabla Usabilidad – Diálogos simples y naturales

|  |  |
| --- | --- |
|  | |
| **Atributo de Calidad** | Usabilidad – Diálogos simples y naturales |
| **Justificación** | La aplicación ATOME cuenta con diálogos simples y claros, indicándole al usuario lo que necesita o puede hacer dentro de la aplicación, además de colores básicos, fáciles de recordar y que anuncian acciones correspondientes al estudio del color. |
| **Fuente** | Usuarios (Estudiantes y Administradores) |
| **Estimulo** | Forma en que interactúa el usuario final sobre el software final. |
| **Artefacto** | Sistema |
| **Ambiente** | Producción |
| **Respuesta** | El usuario deberá poder navegar sin problemas en el sistema y sin la ayuda de algún manual |

### Hablar el lenguaje del usuario

Tabla Usabilidad – Hablar el lenguaje del usuario

|  |  |
| --- | --- |
|  | |
| **Atributo de Calidad** | Usabilidad – Hablar el lenguaje del usuario |
| **Justificación** | La aplicación ATOME cuenta diálogos en lenguaje donde se tutea con el usuario, generando un ambiente de confianza con él, para que se sienta cómodo y facilite la comprensión de las acciones a realizar dentro de la aplicación. |
| **Fuente** | Usuarios (Estudiantes y Administradores) |
| **Estimulo** | Forma en que interactúa el usuario final sobre el software final. |
| **Artefacto** | Sistema |
| **Ambiente** | Producción |
| **Respuesta** | El usuario deberá poder navegar sin problemas en el sistema y sin la ayuda de algún manual |

### Proveer salidas marcadas claramente

Tabla Usabilidad – Proveer salidas marcadas claramente

|  |  |
| --- | --- |
|  | |
| **Atributo de Calidad** | Usabilidad – Proveer salidas marcadas claramente |
| **Justificación** | La aplicación ATOME cuenta con botones que indican acciones de salir de la aplicación, cancelar una acción, ingresar a ella, guardar o modificar datos. |
| **Fuente** | Usuarios (Estudiantes y Administradores) |
| **Estimulo** | Forma en que interactúa el usuario final sobre el software final. |
| **Artefacto** | Sistema |
| **Ambiente** | Producción |
| **Respuesta** | El usuario deberá poder navegar sin problemas en el sistema y sin la ayuda de algún manual |

### Dar buenos mensajes de error

Tabla Usabilidad – Dar buenos mensajes de error

|  |  |
| --- | --- |
|  | |
| **Atributo de Calidad** | Usabilidad – Dar buenos mensajes de error |
| **Justificación** | La aplicación ATOME cuenta con notificaciones que le permiten identificar al usuario de un error o una falta de acciones por parte de él, para llevar a cabo la acción deseada, le sistema le indica puntualmente en que falló. |
| **Fuente** | Usuarios (Estudiantes y Administradores) |
| **Estimulo** | Forma en que interactúa el usuario final sobre el software final. |
| **Artefacto** | Sistema |
| **Ambiente** | Producción |
| **Respuesta** | El usuario deberá poder navegar sin problemas en el sistema y sin la ayuda de algún manual |

### Prevenir y manejar errores

Tabla Usabilidad – Dar buenos mensajes de error

|  |  |
| --- | --- |
|  | |
| **Atributo de Calidad** | Usabilidad – Dar buenos mensajes de error |
| **Justificación** | La aplicación ATOME cuenta con un manejo adecuado de cada una de las validaciones propias de la gestión de datos como las generadas por la aplicación. |
| **Fuente** | Usuarios (Estudiantes y Administradores) |
| **Estimulo** | Forma en que interactúa el usuario final sobre el software final. |
| **Artefacto** | Sistema |
| **Ambiente** | Producción |
| **Respuesta** | El usuario deberá poder navegar sin problemas en el sistema y sin la ayuda de algún manual |

1. Ionic: Ionic es un framework gratuito y open source para desarrollar aplicaciones híbridas multiplataforma que utiliza HTML5, CSS (generado por SASS) y Cordova como base. Es uno de los framework del momento por utilizar AngularJS para gestionar las aplicaciones, lo que asegura aplicaciones rápidas y escalables.  [↑](#footnote-ref-1)
2. Front-End: Son todas aquellas tecnologías que corren del lado del cliente, es decir, todas aquellas tecnologías que corren del lado del navegador web, generalizándose más que nada en tres lenguajes, HTML , CSS Y JavaScript, la persona encargada del front-end, se dedica solo a estas tres tecnologías. [↑](#footnote-ref-2)
3. Back-End: es aquel que se encuentra del lado del servidor, es decir, esta persona se encarga de lenguajes como PHP, Python, .Net, Java, etc, es aquel que se encarga de interactuar con bases de datos, verificar manejos de sesiones de usuarios, montar la página en un servidor, y desde este “servir” todas las vistas que el Front-End crea, es decir, uno como back-end se encarga más que nada de la manipulación de los datos. [↑](#footnote-ref-3)
4. Firebase: Servicio capaz de proveernos de un Backend en la nube con una fuente de datos en tiempo real y librerías para acceder a ella desde aplicaciones web, iOS y Android. [↑](#footnote-ref-4)
5. No SQL: es un término que describe las bases de datos no relacionales de alto desempeño. Las bases de datos No-SQL utilizan varios modelos de datos, incluidos los de documentos, gráficos, claves-valores y columnas. [↑](#footnote-ref-5)
6. JWT: es un método de autenticación mediante tokens. [↑](#footnote-ref-6)
7. Cloud Messagin: es un servicio gratuito que nos permite enviar información desde nuestro servidor a un dispositivo Android. [↑](#footnote-ref-7)
8. Cordova: Es un framework para el desarrollo de aplicaciones móviles propiedad de Adobe Systems que permite a los programadores desarrollar aplicaciones para dispositivos móviles utilizando herramientas web genéricas como JavaScript, HTML5 y CSS3, resultando aplicaciones híbridas. [↑](#footnote-ref-8)
9. AngularJs: Es un proyecto de código abierto, realizado en Javascript que contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web y propone una serie de patrones de diseño para llevarlas a cabo. [↑](#footnote-ref-9)
10. TypeScript: es un lenguaje de programación de código abierto desarrollado y presentado por Microsoft hace unos tres años. Es un superconjunto de JavaScript que esencialmente añade capacidades de POO como es el tipado estático y objetos basados en clases. [↑](#footnote-ref-10)
11. Requerimientos no funcionales: un requisito que sabe bien y especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales. [↑](#footnote-ref-11)