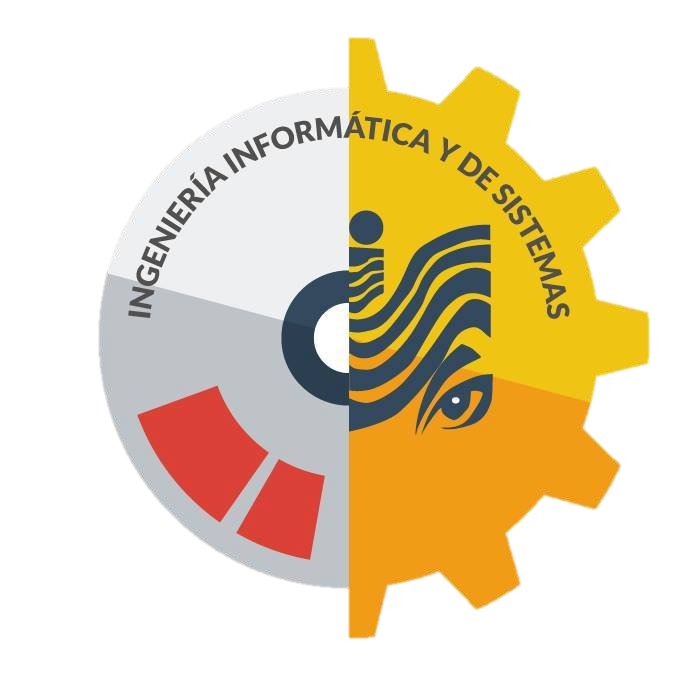


Universidad Nacional de San Antonio Abad del Cusco

**Ingeniería Informática y de Sistemas**

Imagen que contiene texto, reina

Descripción generada con confianza alta

Temas: CRUD Sistema Tutorias  
Curso: Desarrollo de Software  
Integrantes**:**   
- Caceres Quispe, Maria Fernanda 151388   
- Cana Apu Orlando 160888  
- Cconislla Medina Anthony Aldair 161756  
- Hinojosa Huarca, Brayan Alexandert 155190  
- Mamani Quinta Michael Antonni 164566  
- Nolasco Suyo, Carlos Antoni 154631  
-Yupanqui Carrillo, Holger Alfredo 160338  
- Quispe Ynga Callañaupa Luis Angel 170439  
- Vargas Quispe, Charly 083222  
Docente**:** Quintanilla Portugal, Roxana Lisette

Índice

[1. Descripción del problema 3](#_Toc83166423)

[2. Identificación de requisitos 3](#_Toc83166424)

[3. Diagramas de caso de uso 4](#_Toc83166425)

[4. Descripción Tutoría 5](#_Toc83166426)

[5. Distribución de trabajo grupal 6](#_Toc83166427)

[5.1 Grupos para front-end, back-end y base de datos 6](#_Toc83166428)

[6. Test 7](#_Toc83166429)

[6.1 Pruebas Unitarias 7](#_Toc83166430)

[6.2 Pruebas Mocks 8](#_Toc83166431)

[6.3 Pruebas de integración 9](#_Toc83166432)

[6.4 Backlog 10](#_Toc83166433)

[7. Cronograma de Actividades 11](#_Toc83166434)

[8. Trabajo individual 12](#_Toc83166435)

# Descripción del problema

Como sabemos la universidad no tiene un sistema que organize las tutorías donde tenemos algunos de estos problemas son los siguientes:

* Tener una gestión de documentos escritos el cual es un método ineficiente, esto provoca una mala gestión al momento de hacer una asignación de tutor.
* La pérdida de tiempo al momento de hacer un papeleo para dicha asignación.

¿Qué debe hacer nuestro sistema de tutorías?

* Optimizar la gestión de asignación y organización de tutorías.
* Optimización de tiempo.
* Facilidad para el tutor al momento de requerir la interacción con un alumno o viceversa
* Una capacitación del uso del sistema para los docentes que lo requiera.

# Identificación de requisitos

Requerimientos funcionales

|  |  |  |  |
| --- | --- | --- | --- |
| CLAVE | NOMBRE | DESCRIPCION | PRIORIDAD |
| RF-01 | Registro alumno | El sistema permite el registro de un alumno | ALTA |
| RF-02 | Registro docente | el sistema permite el registro de un alumno | ALTA |
| RF-03 | Modificación de datos (docente, alumno) | El sistema permite la modificación de datos únicamente por el personal autorizado | ALTA |
| RF-04 | Eliminar tutor | El sistema permite la eliminación de datos únicamente por el personal autorizado | ALTA |
| RF-05 | Eliminar Alumno | El sistema permite la eliminación de datos únicamente por el personal autorizado | ALTA |

Requerimientos no funcionales

Usabilidad

|  |  |  |  |
| --- | --- | --- | --- |
| CLAVE | NOMBRE | DESCRIPCION | PRIORIDAD |
| RNF-01 | Aprendizaje del sistema | Un usuario inexperto no tardará más de 2 horas en tener dominio total del  sistema. | ALTA |
| RNF-02 | Mensajes de error | El sistema desplegará advertencias y mensajes de error intuitivos. | ALTA |

Desempeño

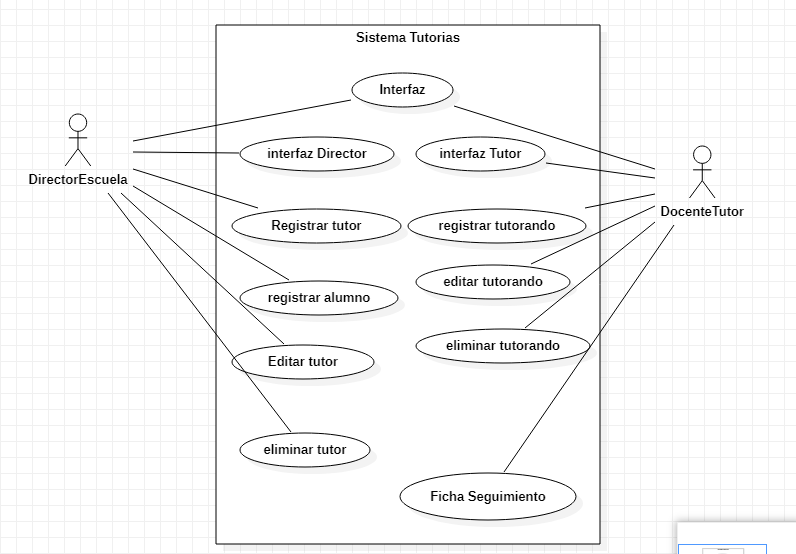
|  |  |  |  |
| --- | --- | --- | --- |
| CLAVE | NOMBRE | DESCRIPCION | PRIORIDAD |
| RNF-03 | Tiempo de  respuesta al operar en el sistema | Se espera que el tiempo de respuesta en el momento de presionar un botón para continuar con el flujo de la  información que no supere los 10 segundos. | ALTA |
| RNF-04 | Escalabilidad | Se espera mantener la escalabilidad del sistema en relación a la concurrencia de  usuarios (cantidad de | ALTA |

Soportabilidad

|  |  |  |  |
| --- | --- | --- | --- |
| CODIGO | NOMBRE | DESCRIPCION | PRIORIDAD |
| RNF-05 | Operatividad del sistema | Cualquier computadora que tenga windows | ALTA |

# Diagramas de caso de uso

* 1. Actores, casos de uso
* Actores
  + DirectorEscuela
  + Tutor
* Casos de uso
  + Interfaz (interfaz Tutor – interfaz DirectorEscuela)
  + Registrar de Alumno
  + Registrar de Docente
  + Registrar Tutorando
  + Buscar por código
  + Modificar datos tutorando
  + Modificar datos Docente
  + Eliminar tutor
  + Eliminar tutorando



# Descripción Tutoría

**Tutoría**

La tutoría es una modalidad de la actividad docente que consiste en un proceso sistemático de orientación y acompañamiento al estudiante de carácter académico y personal durante su estancia en la Universidad, que se concretiza en la atención individualizada que brinda un profesor-tutor a un alumno o a un grupo de alumnos.

**Función del tutor**

* Promueve en el estudiante el mejoramiento de su desempeño académico.
* Estimula su capacidad de aprender a aprender.
* Fomenta su capacidad crítica y creadora.
* Orienta al alumno en los problemas académicos que surjan durante el proceso formativo (dificultades en el aprendizaje; en las relaciones profesor - alumno; entre alumnos; problemáticas personales, familiares, etc.)

**Requisitos para ser tutor**

* Contar con la categoría de nombrado
* Tener el número de horas lectivas necesarias

# 5. Distribución de trabajo grupal

## 5.1 Grupos para front-end, back-end y base de datos

primero se dividió en dos grupos 4 encargados de front-end, 4 encargados de back-end y un encargado de migración de datos:

**Encargados de front-end**

Los encargados de este trabajo se encargaron de las siguientes tablas:

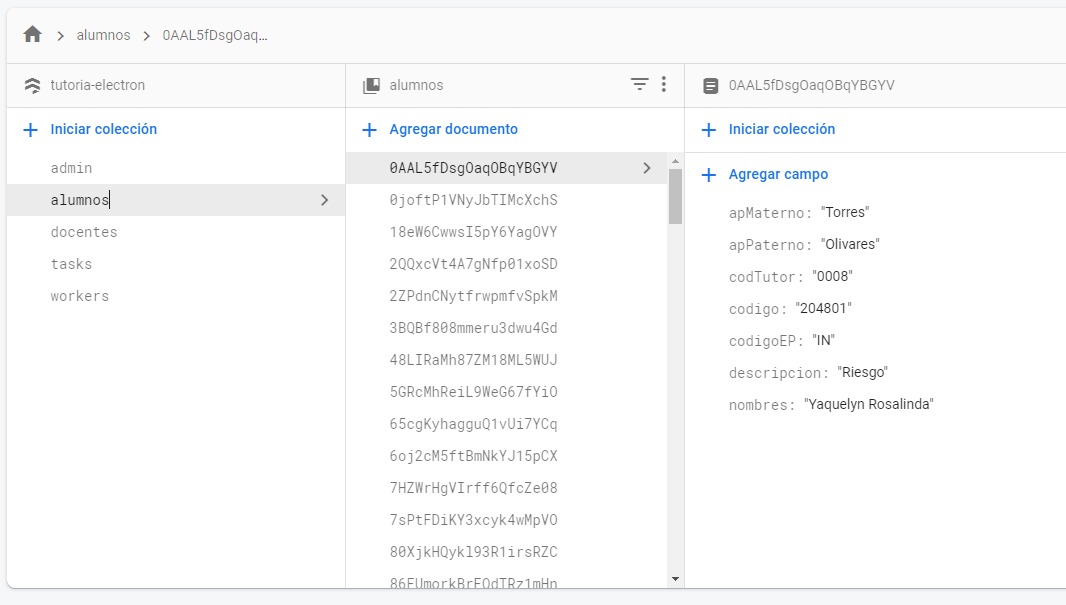
* Tabla alumno
* Tabla docente
* Tabla tutorando
* Tabla ficha seguimiento

**Encargados de back-end**

Los encargados de este trabajo se encargaron de colocar la interfaz grafica necesaria para su funcionamiento y visualización correspondiente, aquí se diseñó el html correspondiente para cada tabla y el login respetivo

**Base de datos**

En el almacenamiento de los datos se optó por usar *FireBase* donde los datos de se almacenan como objetos JSON. La base de datos puede conceptualizarse como un árbol JSON alojado en la nube. A diferencia de una base de datos de SQL, no hay tablas ni registros. Cuando le agregas datos al árbol JSON, estos se convierten en un nodo de la estructura JSON existente con una clave asociada. Puedes entregar tus propias claves, como ID de usuario o nombres semánticos, o también puedes obtenerlas mediante el método push().



**Encargado de migración de datos**

La persona encargada realizo la migración correspondiente a alumnos y docentes correspondientes a la escuela profesional en el cloud que se obtendría en un futuro.

**Cronograma de actividades**

Se procedió a realizar un cronograma de actividades utilizando el Microsoft Proyect con fechas determinadas y con la persona encargada, también se utilizó la plataforma de Tello para la asignación de tareas semanales correspondiente del proyecto y todo esto con el encargo correspondiente

# Test

## **6.1 Pruebas Unitarias**

Para estas pruebas se utiizo Jest donde Jest es un marco de pruebas de Javascript creado por Facebook.

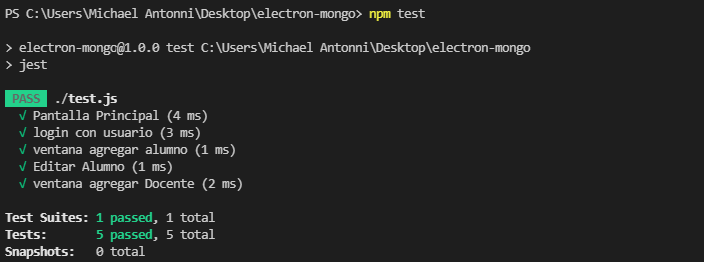
Está diseñado principalmente para aplicaciones basadas en React (que también está desarrollada por Facebook), pero podría usarse para escribir escenarios de automatización para cualquier base de código basada en Javascript.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nro** | **Test Case** | **Input** | **Expected**  **Output** | **Actual Output** | **Remarks** |
| **1** | Agregar Alumno nuevo | Ingresamos todos los datos que nos solicitan | Agregue correctamente al alumno | Agregar correctamente al alumno | Pass |
| **2** | Agregar Alumno vacío | No llenamos ningun parametro | Mensaje que muestre ”No lleno los parámetros requeridos” | Agrega al alumno pero con parámetros vacíos | No Pass |
| **3** | Agregar Alumno con datos faltantes | No ingresamos todos los datos | Mensaje que muestre qué parámetros no digitamos | Agrega al alumno pero llena los datos faltantes conv acio | No Pass |
| **4** | Agregar Alumno que ya existe | Ingresamos datos que se encuentran en la BD | Mensaje que indique “El alumno ya existe” | Agrega al alumno lo cual genera duplicados | No pass |

Se hizo pruebas para:

* la pantalla principal
* login con el ususario 111111 pass 111111
* ventana Registrar Alumno
* ventana Registrar docente
* botón de editar alumno

donde tenemos los resultados siguientes:



## **6.2 Pruebas Mocks**

Para este caso tomamos el método de agregar alumnio por el momento seguimos usando el programa localmente

Hize uso de las pruebas unitarias solo que esta vez utilizando mock que tiene jest

Una de las principales características que diferencia a Jest de otros frameworks de testing es su capacidad para simular o *“hacer mocking”* sobre las dependencias de los módulos que desarrollamos o empleamos en nuestras aplicaciones.

El principal beneficio que proporciona esta técnica y que forma parte de la filosofía de Jest es que, cuando implementamos los tests de un determinado módulo, este debe probarse de manera totalmente aislada del resto de dependencias que se hayan definido para su correcto funcionamiento. Dicho aislamiento lo conseguimos simulando la respuesta de estas dependencias.

Primero creamos la carpeta donde se hara los mocks:

****

## **6.3 Pruebas de integración**

Es una estrategia de prueba en la que todos los componentes individuales se prueban en grupo. Las pruebas integradas intentan replicar la experiencia del usuario ejecutando las pruebas en un navegador real. Esto es considerablemente más lento que las pruebas funcionales y las pruebas unitarias porque cada conjunto de pruebas se ejecuta en un navegador en vivo.

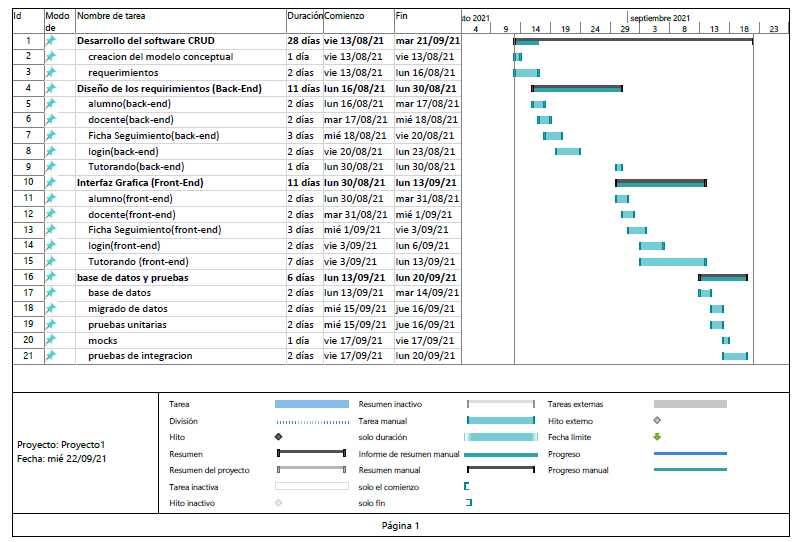
|  |  |  |  |
| --- | --- | --- | --- |
| **Nro** | **Test Case** | **Expected output** | **Real Output** |
| 1 | Verificar la conexión entre la interfaz con la pestaña estudiantes | Al hacer el click en botón lista de alumnos mostrar la relación de estudiantes | Muestra una lista con los alumnos pero no sus correspondientes tutores |
| 2 | Verificar la conexión entre la interfaz con la pestaña agregar docentes | Al hacer el click en botón agregar docentes mostrar el form de rellenado de docentes | Muestra correctamente los espacios con las solicitudes de datos de crear docentes |
| 3 | Verificar la conexión con la pestaña agregar tutores | Al hacer click en el botón agregar tutores mostrara la asignación de un docente a su respectivo tutorando | Muestra los datos del docente y el código del alumno |
| 4 | Verificar la conexión la Pestaña Ficha de Tutorías | Al hacer click en un ficha esta muestra la interfaz de rellenado de una ficha de tutoría | Muestra el form del llenado de la ficha de tutorías |
| 5 | Verificar la conexión con la pestaña de tutorando | Al realizar la petición de un tutorando se muestra los datos de este | Muestra los datos correspondientes al alumno con su ficha de seguimiento |

## **6.4 Backlog**

Agrupacion por moscow

|  |  |  |
| --- | --- | --- |
| Nro Tarea | Tarea | Prioridad MoSCow |
| 1 | Registrar en el sistema a los docentes que serán tutores | M |
| 2 | Asignar a un Estudiante su tutor correspondiente | M |
| 3 | Registrar Ficha de Tutoría de un Estudiante | M |
| 4 | Modificar Datos Estudiante | C |
| 5 | Modificar Datos Docentes | C |
| 6 | Agregar o Eliminar Docentes | S |
| 7 | Agregar o Eliminar Tutores | S |
| 8 | Visualizar la lista de Tutores | M |
| 9 | Visualizar la lista de Estudiantes | M |
| 10 | Agregar o Eliminar Estudiantes | M |

# 7. Cronograma de Actividades



# Trabajo individual

|  |  |
| --- | --- |
| **Integrantes** | **Aportaciones al proyecto** |
| Cana Apu Orlando | * Docente js |
| Cconislla Medina Anthony Aldair | * Docente js * alumno js * ficha seguimiento js * tutorando js * login * creación de la base de datos * Documentacion |
| Caceres Quispe Maria Fernanda | * Html docente |
| Hinijosa Huarca Brayan Alexander | * Login * Pruebas de integración * Documentación * Html Tutorando |
| Mamani Quinta Michael Antonni | * Alumno js * Pruebas unitarias * Mocks * Documentacion * Readme (github) |
| Nolazco Suyo Carlos Antoni | * Html alumno * Html Tutorando * Documentación |
| Yupanqui Carrillo Holger Alfredo | * Migrado de datos a Firebase |
| Quispe Ynga Callañaupa Luis Angel | * Html ficha de seguimiento |
| Vargas Quispe Charly | * Html ficha de seguimiento |

Trabajo en GitHub



