

UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO  
FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA  
ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS



# Informe Final

**Asignatura:** Ing. de Software I

**Docente:** Roxana Lissette Quintanilla Portugal

**Integrantes:**

- 170429 CONDORI LOPEZ, Juan Carlos
- 174442 ESCOBEDO MESCCO, Angie
- 171258 ESPEJO FRANCO, Melissa
- 170432 GUTIERREZ DAZA, Gonzalo
- 150394 HUAMAN GUEVARA, Alexander Javier
- 171915 NINANTAY DIAZ, Mileydy
- 171570 RAMOS ALVAREZ, Edgar
- 171805 ROJAS SOTO, Claudia Luz

# Cusco - Perú

## 2021

## 1. Filtrar las tareas que sean realizab

Nº	Rastr o	Tarea
1	T1	Convocar a concurso de estudiantes
2	T2	Filtrar a los docentes que serán tutores
3	T3	Obtener lista de matriculados
4	T4	Seleccionar a los alumnos que hayan desaprobado dos veces un curso
5	T5	Obtener avance curricular de estudiantes
6	T6	Recibir informes de tutores
7	T7	Solicitar horarios disponibles de los tutores
8	T8	Seleccionar lugar de tutorías
9	T9	Filtrar los datos de los estudiantes de Ing. Informática
10	T10	Entrar al concurso
11	T11	Asistir a tutorías
12	T12	Registrar horarios disponibles
13	T13	Registrar informes de estado situacional de estudiantes con R.A.
14	T14	Registrar fichas de tutoría
15	T15	Registrar sesiones de tutoría
16	T16	Consultar informes de tutoría
17	T17	Registrar informe semestral de tutoría
18	T18	Filtrar a los estudiantes que puedan ser ayudante de tutoría
19	T19	Completar registro de horarios de tutoría
20	T20	Registrar estudiantes
21	T21	Registrar docentes
22	T22	Actualizar datos personales
23	T23	Recuperar lista de ingresantes de Ing. Informática (cada semestre)
24	T24	Llenar formulario para cada alumno
25	T25	Cargar documento .csv (por lotes) de alumnos
26	T26	Llenar formulario para cada docente
27	T27	Cargar documento .csv (por lotes) de docentes
28	T28	Asignar tutor y tutorandos
23	T29	Recuperar lista de docentes de Ing. Informática
29	T30	<b>Confidencialidad Fichas de Tutoría</b>
30	T31	<b>Confidencialidad Inicio de Sesión</b>

### Tareas realizables:

Nº	Rastr o	Tarea
1	T2	Filtrar a los docentes que serán tutores
2	T13	Registrar informes de estado situacional de estudiantes con R.A.
3	T14	Registrar fichas de tutoría
4	T15	Registrar sesiones de tutoría
5	T16	Consultar informes de tutoría
6	T17	Registrar informe semestral de tutoría
7	T18	Filtrar a los estudiantes que puedan ser ayudante de tutoría
8	T19	Completar registro de horarios de tutoría
9	T20	Registrar estudiantes
10	T21	Registrar docentes
11	T22	Actualizar datos personales
12	T28	Asignar tutor y tutorandos
13	T30	<b>Confidencialidad Fichas de Tutoría</b>
14	T31	<b>Confidencialidad Inicio de Sesión</b>

## 2. Priorizar de acuerdo a alguna técnica de priorización de requisitos

### PRIORIZACIÓN MOSCOW

Usamos esta priorización para segmentar las tareas en grupos y darle un nuevo enfoque al orden de la lista que tenemos.

Tarea	MOSCO W	ORDEN
Filtrar a los docentes que serán tutores	M	3
Registrar informes de estado situacional de estudiantes con R.A.	S	10
Registrar fichas de tutoría	M	6
Registrar sesiones de tutoría	M	8
Consultar informes de tutoría	C	14
Registrar informe semestral de tutoría	M	9
Filtrar a los estudiantes que puedan ser ayudantes de tutoría	S	12

Completar registro de horarios de tutoría	C	13
Registrar estudiantes	M	1
Registrar docentes	M	2
Actualizar datos personales	S	11
Asignar tutor y tutorandos	M	5
<b>Confidencialidad Fichas de Tutoría</b>	M	7
<b>Confidencialidad Inicio de Sesión</b>	M	4

### Paso 1: Orden de prioridad

Donde el primero de la lista es el más prioritario, y el último el menos prioritario.

Tarea	ORDEN
Registrar estudiantes	1
Registrar docentes	2
Filtrar a los docentes que serán tutores	3
<b>Confidencialidad Inicio de Sesión</b>	4
Asignar tutor y tutorandos	5
Registrar fichas de tutoría	6
<b>Confidencialidad Fichas de Tutoría</b>	7
Registrar sesiones de tutoría	8
Registrar informe semestral de tutoría	9
Registrar informes de estado situacional de estudiantes con R.A.	10
Actualizar datos personales	11
Filtrar a los estudiantes que puedan ser ayudantes de tutoría	12
Completar registro de horarios de tutoría	13
Consultar informes de tutoría	14

## **Binary Search Tree**

### **Construcción:**

- Poner todos los requisitos en una pila.
- Tomar un requisito y póngalo como nodo raíz.
- Tomar otro requisito y compárelo con el nodo raíz.
- Si el requisito es menos importante que el nodo raíz, comparar con el nodo hijo izquierdo. Si el requisito es más importante que el nodo raíz, comparar con el nodo hijo correcto. Si el nodo no tiene ningún nodo hijo apropiado, insertar el nuevo requisito como el nuevo nodo hijo a la derecha o izquierda, dependiendo de si el requisito es más o menos importante.
- Repetir los pasos 3 a 4 hasta que se hayan comparado todos los requisitos y se hayan insertado en la BST.
- Para fines de presentación, recorrer todo el BST en orden y colocar los requisitos en una lista, con el requisito menos importante al final de la lista y el requisito más importante al principio de la lista

### **Recomendaciones:**

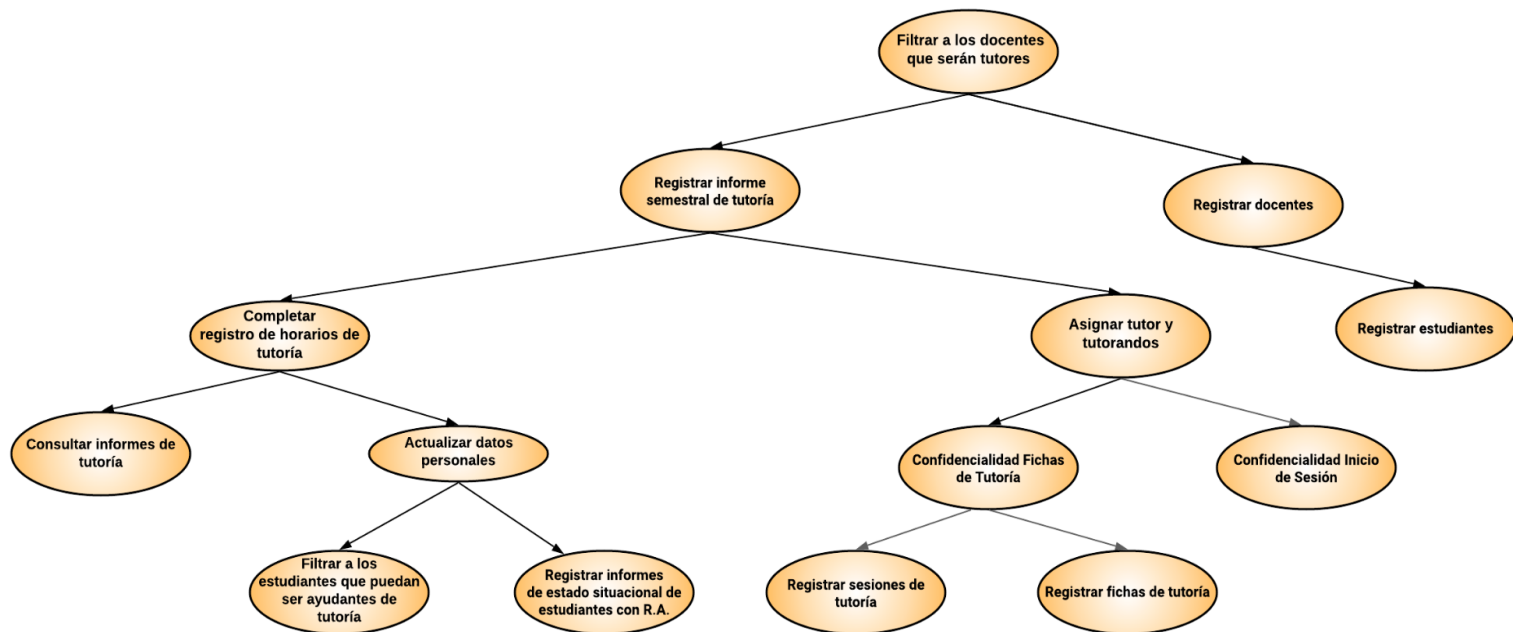
- Como preparación, describa los requisitos del candidato.
- Como ejecución, seleccione los requisitos uno a la vez y cree un árbol de búsqueda binario.
- Como presentación, recorra el árbol de búsqueda binaria en inorden y agreguelos a una lista. Los requisitos que tienen la prioridad más baja aparecen primero en la lista.

### **Paso 2: Pila de Requisitos**

Registrar estudiantes
Registrar docentes
Filtrar a los docentes que serán tutores
<b>Confidencialidad Inicio de Sesión</b>
Asignar tutor y tutorandos
Registrar fichas de tutoría
<b>Confidencialidad Fichas de Tutoría</b>
Registrar sesiones de tutoría
Registrar informe semestral de tutoría
Registrar informes de estado situacional de estudiantes con R.A.
Actualizar datos personales
Filtrar a los estudiantes que puedan ser ayudantes de tutoría

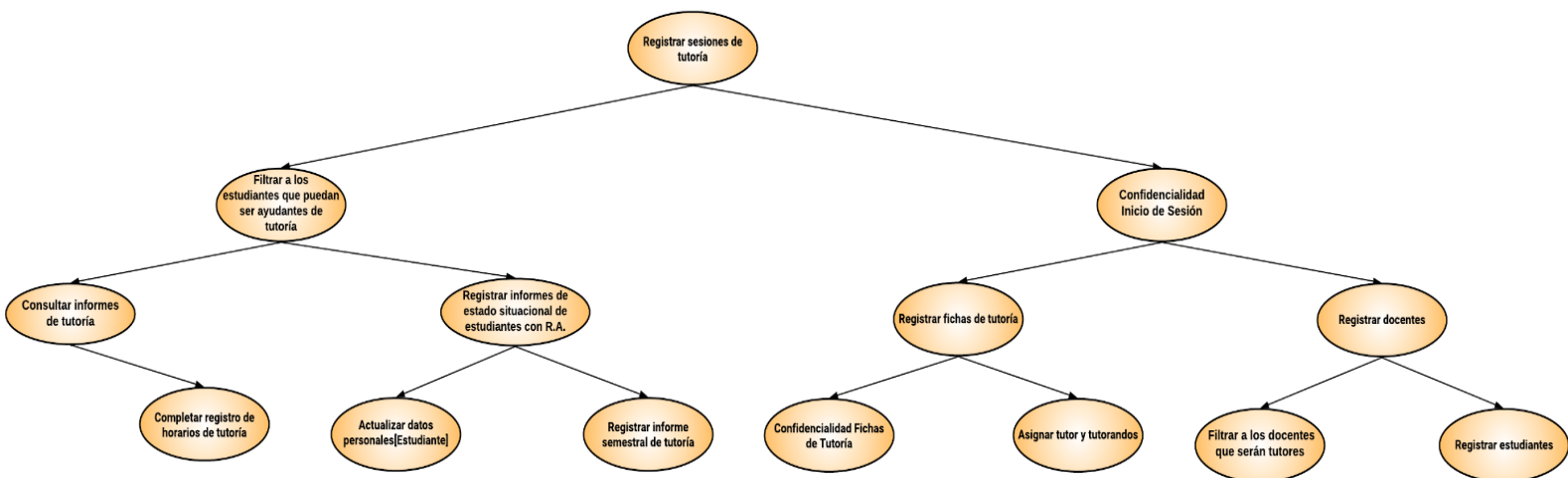
Completar registro de horarios de tutoría

Consultar informes de tutoría



### Paso 3: Construcción del BTS

### Paso 4: Balanceo del BTS



### Paso 5: Creación del Backlog

Para la creación del backlog se siguió el orden en que las tareas fueron priorizadas en el BTS (lo que indica el orden que deben ser resueltas), además se utilizó la priorización MOSCOW para agruparlas en 3 grupos.

<b>PRIORIDAD ALTA (DEL 1 AL 9)</b>	<b>PRIORIDAD MEDIA (10 AL 12)</b>	<b>PRIORIDAD BAJA (13 AL 14)</b>
1. Registrar estudiantes	10. Registrar informes de estado situacional de estudiantes con R.A.	13. Completar registro de horarios de tutoría
2. Registrar docentes	11. Actualizar datos personales	14. Consultar informes de tutoría
3. Filtrar a los docentes que serán tutores	12. Filtrar a los estudiantes que puedan ser ayudantes de tutoría	
4. Confidencialidad Inicio de Sesión		
5. Asignar tutor y tutorandos		
6. Registrar fichas de tutoría		
7. Confidencialidad Fichas de Tutoría		
8. Registrar sesiones de tutoría		
9. Registrar informe semestral de tutoría		

### 3. Especificación de Requisitos

#### Registrar estudiantes

<b>Requerimiento</b>	R1	<b>Tipo Requerimiento</b>	Funcional	<b>Eventos / Goal</b>	G12
<b>Descripción:</b>	El sistema permite registrar a un nuevo o nuevos estudiantes.				
<b>Fundamentos:</b>	Cada semestre existen nuevos estudiantes ingresantes a la carrera profesional, por lo que se debe de agregar a estos nuevos estudiantes a la base de datos del sistema de tutorías para que posteriormente se les sea asignado un docente tutor.				
<b>Actor:</b>	Director de Escuela				
<b>Criterio:</b>	El director de escuela podrá realizar el registro por lote (varios estudiantes) subiendo un archivo de excel en el sistema, o por unidad (un estudiante) registrando a través de un formulario los datos del nuevo estudiante.				
<b>Prioridad:</b>	Alta		<b>Conflictos</b>	No posee	

#### Registrar docentes

<b>Requerimiento</b>	R2	<b>Tipo Requerimiento</b>	Funcional	<b>Eventos / Goal</b>	G13
<b>Descripción:</b>	El sistema permite registrar a un nuevo docente.				
<b>Fundamentos:</b>	Para poder realizar el filtrado de docentes tutores, se necesita tener un listado de los docentes, y además cada cierto tiempo hay nuevos docentes que forman parte de la escuela profesional.				
<b>Actor:</b>	Director de Escuela				
<b>Criterio:</b>	El director de escuela podrá realizar el registro de los datos del nuevo docente a través del llenado de un formulario.				
<b>Prioridad:</b>	Alta		<b>Conflictos</b>	No posee	



**Filtrar a los docentes que serán tutores**

<b>Requerimiento</b>	R3	<b>Tipo Requerimiento</b>	Funcional	<b>Eventos / Goal</b>	G2
<b>Descripción:</b>	El sistema permitirá filtrar a los docentes que serán tutores.				
<b>Fundamentos:</b>	Tener el registro de docentes que serán tutores en el semestre lectivo				
<b>Actor:</b>	Director de Escuela				
<b>Criterio:</b>	El sistema tiene que tener la capacidad para poder filtrar a cada uno de los docentes que tengan los requisitos para poder ser tutores (docentes a tiempo completo), para poder asignarles una determinada carga de tutorandos, y programar su horario de tutoría.				
<b>Prioridad:</b>	Alta		<b>Conflictos</b>	No posee	

**Confidencialidad Inicio de Sesión**

<b>Requerimiento</b>	R4	<b>Tipo Requerimiento</b>	No Funcional	<b>Eventos / Goal</b>	
<b>Descripción:</b>	El sistema restringirá el acceso a sus distintas funcionalidades dependiendo del actor				
<b>Fundamentos:</b>	No todas las funcionalidades corresponden a todos los actores				
<b>Actor:</b>	Todos				
<b>Criterio:</b>	El sistema tendrá la capacidad de identificar el tipo de actor que está intentando ingresar y autenticará su inicio de sesión. Luego, le mostrará las funcionalidades que le corresponden.				
<b>Prioridad:</b>	Alta		<b>Conflictos</b>	No posee	

**Asignar tutor y tutorandos**

<b>Requerimiento</b>	R5	<b>Tipo Requerimiento</b>	Funcional	<b>Eventos / Goal</b>	
<b>Descripción:</b>	El sistema registrará la asignación de uno o varios tutorandos a un tutor correspondiente.				

<b>Fundamentos:</b>	Para lograr que cada tutor tenga sus tutorandos en el semestre académico correspondiente.		
<b>Actor:</b>	Director de Escuela		
<b>Criterio:</b>	Servirá para realizar la asignación de una determinada carga de tutorandos. Sabiendo que un tutorando puede tener un tutor en el semestre académico. Todos los tutores tienen la misma cantidad de tutorandos y de diferentes años de ingreso.		
<b>Prioridad:</b>	Alta	<b>Conflictos</b>	No posee

#### Registrar fichas de tutoría

<b>Requerimiento</b>	R6	<b>Tipo Requerimiento</b>	Funcional	<b>Eventos / Goal</b>	G4
<b>Descripción:</b>	El sistema permitirá a cualquier docente tutor, registrar la ficha de tutoría de cada uno de sus tutorandos.				
<b>Fundamentos:</b>	El docente tutor requiere llevar el control de sus fichas de tutoría para registrar sus sesiones correspondientes.				
<b>Actor:</b>	Tutor				
<b>Criterio:</b>	Servirá para llevar el control de las sesiones de tutoría de cada uno de los tutorandos asignados a través de una ficha de tutoría. Se tendrá en cuenta el número telefonico de referencia y persona de referencia.				
<b>Prioridad:</b>	Alta	<b>Conflictos</b>	No posee		

#### Confidencialidad Fichas de Tutoría

<b>Requerimiento</b>	R7	<b>Tipo Requerimiento</b>	No Funcional	<b>Eventos / Goal</b>	
<b>Descripción:</b>	El sistema restringirá el acceso a los datos de la ficha de tutoría				
<b>Fundamentos:</b>	En algunas ocasiones la información brindada por los estudiantes es de carácter confidencial (que solo sea conocida por él y su tutor)				
<b>Actor:</b>	Tutor				
<b>Criterio:</b>	La disponibilidad de la información brindada por el estudiante dependerá de él. En caso no acepte, se aplicará el algoritmo RSA, para encriptar la información de las observaciones anotadas por el docente tutor. Cualquier información de carácter confidencial estará contenida en este campo de la ficha. Mientras que el resto de información estará disponible para el coordinador de tutoría. Se requerirá que el docente vuelva a ingresar su contraseña para poder ver las fichas de tutoría que				

	almacenó.		
<b>Prioridad:</b>	Alta	<b>Conflictos</b>	No posee

#### Registrar sesiones de tutoría

<b>Requerimiento</b>	R8	<b>Tipo Requerimiento</b>	No Funcional	<b>Eventos / Goal</b>	
<b>Descripción:</b>	El sistema permitirá a cualquier docente tutor, llenar la sesión de tutoría de las fichas asignadas de tutoría de cada uno de sus tutorandos.				
<b>Fundamentos:</b>	El docente tutor requiere llevar el control de sus sesiones de tutoría de los tutorandos correspondientes.				
<b>Actor:</b>	Tutor				
<b>Criterio:</b>	Servirá para realizar el control de las sesiones de tutoría de cada uno de los tutorandos asignados a través de una ficha de tutoría. Esta se podrá registrar solo una vez de acuerdo a los datos del tutorando a cargo, y esta ficha sólo podrá actualizarse 3 veces durante el semestre y excepcionalmente más, dependiendo del caso. Se tendrá en cuenta el tipo de tutoría dada, el semestre, la descripción de la actividad, la fecha, algún tipo de observación y la referencia.				
<b>Prioridad:</b>	Alta	<b>Conflictos</b>	No posee		

#### Registrar informe semestral de tutoría

<b>Requerimiento</b>	R9	<b>Tipo Requerimiento</b>	Funcional	<b>Eventos / Goal</b>	G4/G9
<b>Descripción:</b>	El sistema permite registrar el informe semestral de tutoría.				
<b>Fundamentos:</b>	El informe semestral de tutoría de cada docente debe estar registrado en el sistema para que el coordinador de tutoría tenga conocimiento sobre cómo fue el desarrollo de las tutorías dentro de un semestre académico.				
<b>Actor:</b>	Tutor				
<b>Criterio:</b>	El tutor tiene la obligación de enviar un informe a la finalización del semestre en el cual debe especificar: un resumen sobre la cantidad de alumnos asistentes a las distintas sesiones de tutoría brindadas durante el semestre, la cantidad de los alumnos que necesitaron más de 3 sesiones, y los alumnos que fueron derivados a otras áreas de la universidad. Este informe solo deberá ser enviado una vez por el docente tutor.				
<b>Prioridad:</b>	Alta	<b>Conflictos</b>	No posee		

**Registrar informes de estado situacional de estudiantes con R.A.**

<b>Requerimiento</b>	R10	<b>Tipo Requerimiento</b>	Funcional	<b>Eventos / Goal</b>	G5
<b>Descripción:</b>	El sistema permite registrar el informe quincenal de la situación de estudiantes con R.A.				
<b>Fundamentos:</b>	El informe quincenal sobre el estado situacional de los estudiantes con R.A es necesario para que el coordinador de tutoría pueda tener un mejor control sobre este y su mejora a lo largo del semestre.				
<b>Actor:</b>	Tutor				
<b>Criterio:</b>	El tutor tiene la obligación de enviar un informe cada 15 días sobre el estado de cada uno de los estudiantes con R.A. indicando cómo ha sido el desarrollo de las sesiones de tutoría y adjuntando breves descripciones sobre el avance de este alumno durante el semestre.				
<b>Prioridad:</b>	Media		<b>Conflictos</b>	No posee	

**Actualizar datos personales**

<b>Requerimiento</b>	R11	<b>Tipo Requerimiento</b>	Funcional	<b>Eventos / Goal</b>	
<b>Descripción:</b>	El sistema permite registrar que un estudiante pueda realizar modificaciones o actualizaciones de sus datos personales.				
<b>Fundamentos:</b>	El tener un registro de los datos personales actualizados de un estudiante es necesario para la realización de las sesiones de tutoría con su docente tutor.				
<b>Actor:</b>	Estudiante				
<b>Criterio:</b>	El estudiante solo podrá actualizar algunos datos personales como número de celular y dirección; por otro lado no podrá modificar datos como su código de estudiante, correo, apellidos o nombres a través de un formulario.				
<b>Prioridad:</b>	Media		<b>Conflictos</b>	No posee	

**Filtrar a los estudiantes que puedan ser ayudantes de tutoría**

<b>Requerimiento</b>	R12	<b>Tipo Requerimiento</b>	Funcional	<b>Eventos / Goal</b>	G1
<b>Descripción:</b>	El sistema permite filtrar a los estudiantes que puedan ser ayudantes de tutoría				

<b>Fundamentos:</b>	El director de escuela requiere el listado de los estudiantes que podrían ser ayudantes de tutoría.		
<b>Actor:</b>	Director de Escuela		
<b>Criterio:</b>	El sistema debe ser capaz de filtrar a alumnos que cumplan el perfil de un estudiante ayudante de tutoría para cada uno de los cursos dictados en la escuela profesional. Uno de los criterios para hacer este filtro son las notas obtenidas en los últimos semestres.		
<b>Prioridad:</b>	Media	<b>Conflictos</b>	No posee

#### Completar registro de horarios de tutoría

<b>Requerimiento</b>	R13	<b>Tipo Requerimiento</b>	Funcional	<b>Eventos / Goal</b>	G7
<b>Descripción:</b>	El sistema permitirá al coordinador de tutoría, llevar un registro de los horarios y lugares de tutoría asignados a cada docente tutor				
<b>Fundamentos:</b>	Tener todo el registro de horarios de tutoría para poder monitorear las sesiones de los tutores				
<b>Actor:</b>	Coordinador de Tutoría				
<b>Criterio:</b>	El sistema recibirá el nombre del docente tutor, la hora de tutoría asignada y la sala meet creada a fin de desarrollar las tutorías. Estos horarios serán previamente establecidos en coordinación con los tutores.				
<b>Prioridad:</b>	Baja	<b>Conflictos</b>	No posee		

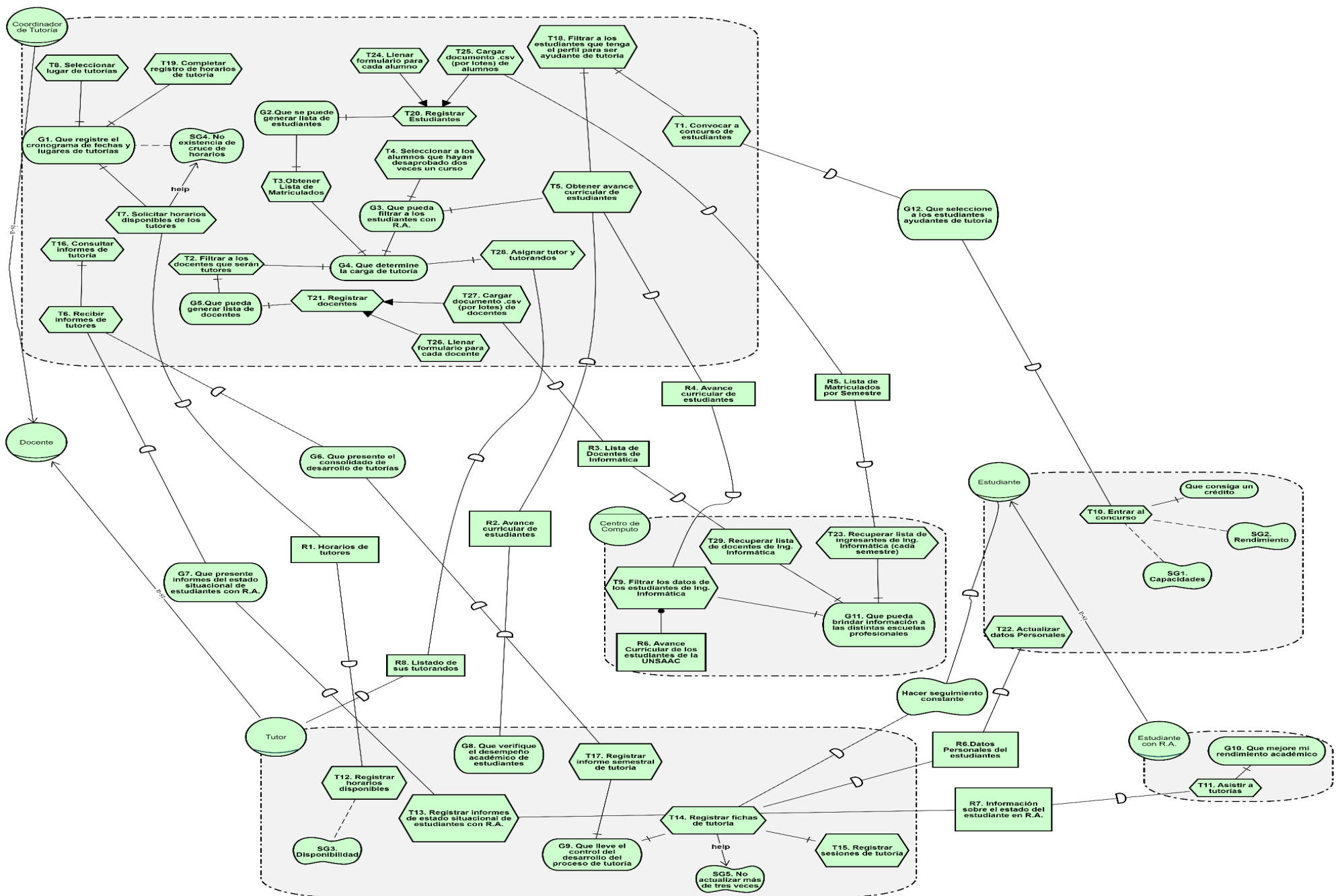
#### Consultar informes de tutoría

<b>Requerimiento</b>	R14	<b>Tipo Requerimiento</b>	Funcional	<b>Eventos / Goal</b>	G5/G9
<b>Descripción:</b>	El sistema permitirá que el coordinador de tutoría pueda acceder a los informes de tutoría.				
<b>Fundamentos:</b>	El coordinador de tutoría requiere consultar los informes enviados por los tutores.				
<b>Actor:</b>	Coordinador de Tutoría				
<b>Criterio:</b>	Los informes serán emitidos por los docentes tutores. Los informes pueden ser: informes semestrales o informes del estado situacional de los alumnos que se presentan cada 15 días. El coordinador tendrá la capacidad de buscar filtrar los informes por docente o por semestre.				
<b>Prioridad:</b>	Baja	<b>Conflictos</b>	No posee		

#### **4. Modelo SR**

Para su mejor visualización, ver:

[https://github.com/Claudiars20/PeruvianTechnologies\\_v2/tree/main/Segundo%20Entregable/Actualizaci%C3%B3n%20Requisitos](https://github.com/Claudiars20/PeruvianTechnologies_v2/tree/main/Segundo%20Entregable/Actualizaci%C3%B3n%20Requisitos)



## 5. Prototipos

Deploy: <https://sistema-tutorias.netlify.app/>

- Filtrar a los docentes que serán tutores

Tutorias



⚡

Notificacion

⚡

Horarios de Tutoria

⚡

Docentes Tutores

⚡

Ayudantes de tutoria

⚡

Estudiantes con Riesgo

⚡

Consultar Informes de tutoria

⚡

Crear Notificacion

Asignar Tutor

⚡

Estudiantes

⚡

Docentes

Perfil

Bienvenido : Usuario

Docentes Tutores

Cerrar Sesión

CodDocente	Nombres y Apellidos	Categoria
1	sample text	sample text
2	sample text	sample text
3	sample text	sample text
4	sample text	sample text
5	sample text	sample text
6	sample text	sample text
7	sample text	sample text
8	sample text	sample text
9	sample text	sample text
10	sample text	sample text

- Registrar informes de estado situacional de estudiantes con R.A.

Tutorias



⚡

Inicio

⚡

Estudiantes a Cargo

⚡

Estudiantes Ayudantes

⚡

Realizar Informe quincenal de tutorados con riesgo academico

⚡

Informe semestral

⚡

Registrar ficha de tutoria

⚡

Registrar Sesión de Tutoria

Obtencion de notas

⚡

Perfil

Bienvenido : Usuario

Informe Quincenal

Cerrar Sesión

Tutorados :  Semestre :

1. Asistencia -Reuniones Programadas

Nro	Fecha	Asistio	Descripcion
1	sample text	sample text	sample text
2	sample text	sample text	sample text

2. Resumen de Reuniones :

3. Dificultades



- Registrar fichas de tutoría


**Tutorías**



Inicio

Estudiantes a Cargo

Estudiantes Ayudantes

Realizar Informe quincenal de tutorados con riesgo académico

Informe semestral

Registrar ficha de tutoría

Registrar Sesión de Tutoría

Obtencion de notas

Perfil

Bienvenido : Usuario

Fichas de Tutoría

Cerrar Sesión

Tutorandos

Search

CodEstudiante	Nombres	ApPaterno	ApMaterno	Persona Referencia	Celular de referencia	
1	sample text	sample text	sample text	sample text	sample text	Editar
2	sample text	sample text	sample text	sample text	sample text	Editar
3	sample text	sample text	sample text	sample text	sample text	Editar
4	sample text	sample text	sample text	sample text	sample text	Editar
5	sample text	sample text	sample text	sample text	sample text	Editar
6	sample text	sample text	sample text	sample text	sample text	Editar
7	sample text	sample text	sample text	sample text	sample text	Editar
8	sample text	sample text	sample text	sample text	sample text	Editar
9	sample text	sample text	sample text	sample text	sample text	Editar
10	sample text	sample text	sample text	sample text	sample text	Editar
11	sample text	sample text	sample text	sample text	sample text	Editar

- Registrar sesión de Tutoría


**Tutorías**



Inicio

Estudiantes a Cargo

Estudiantes Ayudantes

Realizar Informe quincenal de tutorados con riesgo académico

Informe semestral

Registrar ficha de tutoría

Registrar Sesión de Tutoría

Obtencion de notas

Perfil

Bienvenido : Usuario

Sesión Tutoría

Cerrar Sesión

Semestre : 
Tutorado:

Fecha	TipoTutoría	Descripción	Referencia	Observaciones
sample text	sample text	sample text	sample text	sample text
sample text	sample text	sample text	sample text	sample text
sample text	sample text	sample text	sample text	sample text
sample text	sample text	sample text	sample text	sample text
sample text	sample text	sample text	sample text	sample text

Nueva Sesión

- Registrar informe semestral de tutoría

**Tutorías**

Inicio  
Estudiantes a Cargo  
Estudiantes Ayudantes  
Realizar Informe quincenal de tutorados con riesgo académico  
Informe semestral  
Registrar ficha de tutoría  
Registrar Sesión de Tutoría  
Obtención de notas  
Perfil

Bienvenido : Usuario
Informe Semestral
Cerrar Sesión

Cuadro Resumen
Semestre : 2020-I

Indicador	Cantidad	Porcentaje
Tutorados total	sample text	sample text
Estudiante s con tutorías realizadas a inicio de semestre	sample text	sample text
Estudiantes con tutorías realizadas a medio semestre	sample text	sample text
Estudiantes con tutorías realizadas a final de sesmtr	sample text	sample text

Cayos de tutoría que implicaron mas de dos sesiones
Tutorado : new value 1
Añadir

Nro	Curso	Estudiante	Detalles
1	sample text	sample text	sample text
2	sample text	sample text	sample text

Cayos de tutoría que implicaron mas de dos sesiones
Tutorado : new value 1
Añadir

Nro	Curso	Estudiante	Detalles
1	sample text	sample text	sample text
2	sample text	sample text	sample text

Enviar

- Filtrar a los estudiantes que puedan ser ayudantes de tutoría

**Tutorías**

Notificación  
Horarios de Tutoría  
Docentes Tutores  
Ayudantes de tutoría  
Estudiantes con Riesgo  
Consultar Informes de tutoría  
Crear Notificación  
Asignar Tutor  
Estudiantes  
Docentes  
Perfil

Bienvenido : Usuario
Ayudantes Tutoría
Cerrar Sesión

Semestre : 2020-I

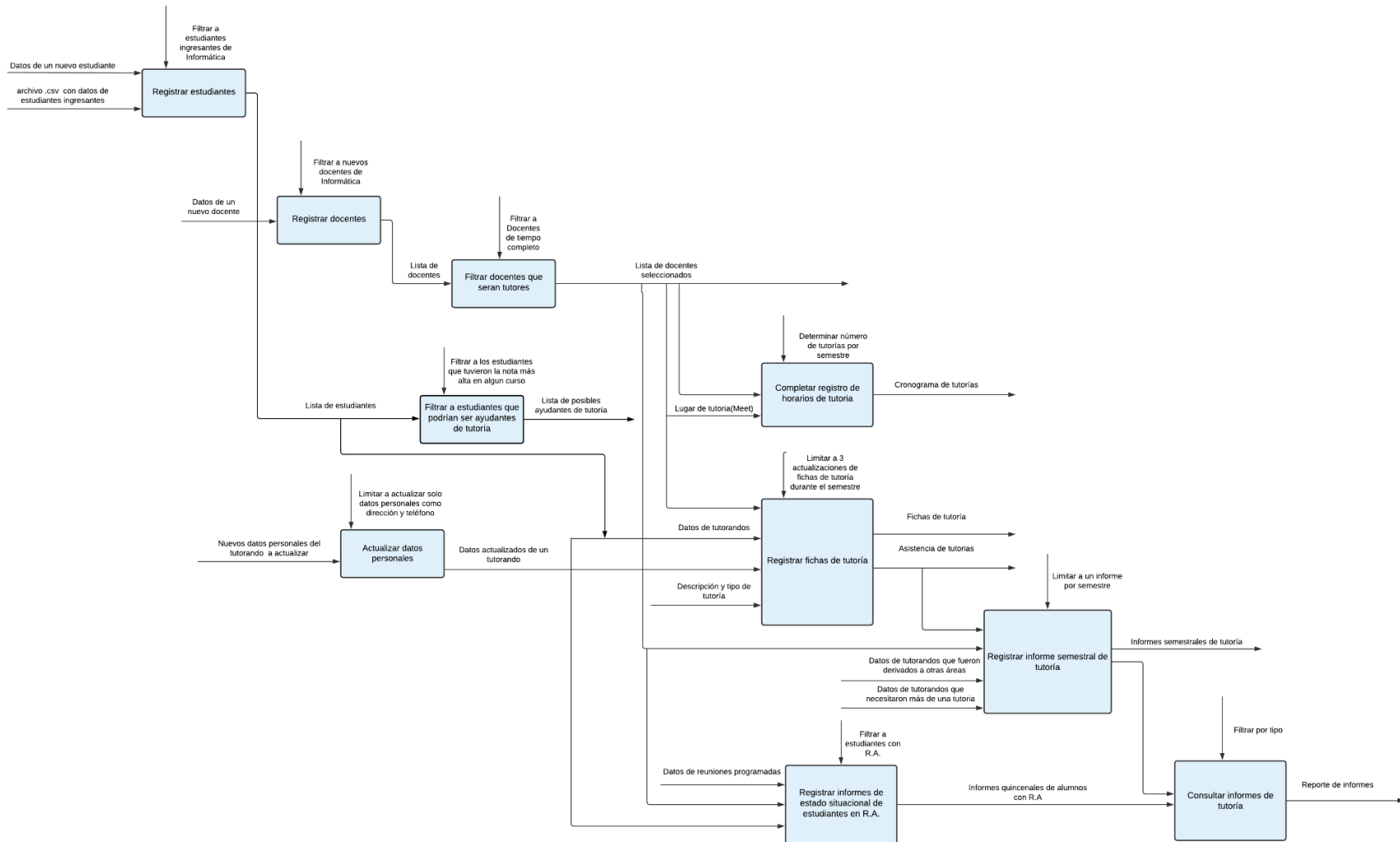
Asignatura: new value 1

Ranking de Estudiantes :
Consultar

Nro	Nombres Apellidos	Categoría	Lista Tutorados
1	sample text	sample text	
2	sample text	sample text	
3	sample text	sample text	
4	sample text	sample text	
5	sample text	sample text	
6	sample text	sample text	
7	sample text	sample text	



## ANEXO: DIAGRAMA SADT



# Code Smells

## ¿Qué son los Code Smells?

Síntomas en el código que ciertos procesos no se hacen de manera correcta y que pueden crear problemas a futuro. Por lo general no son problemas de programación, no son técnicamente incorrectos y quizás el programa funciona correctamente, sin embargo si indican deficiencia en el diseño con un desarrollo más lento, aumentando el riesgo de errores y fallos en el futuro.

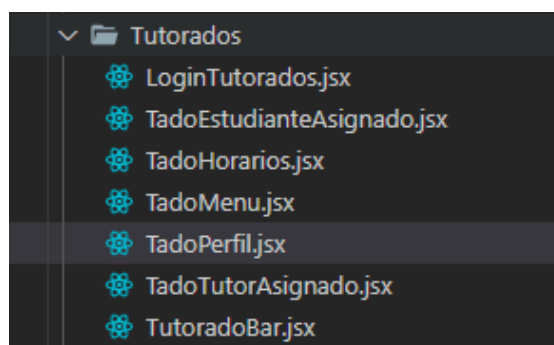


JavaScript no es un lenguaje orientado a objetos del todo, no fue diseñado para serlo, la noción de clases no le es aplicable en absoluto. Si bien todo en JS es de hecho un objeto, estos objetos son diferentes de los de Java o C #.

**Programación reactiva, o Reactive Programming**, es un paradigma enfocado en el trabajo con flujos de datos finitos o infinitos de manera asíncrona, permitiendo que estos datos se propaguen generando cambios en la aplicación, es decir, “reaccionan” a los datos ejecutando una serie de eventos.

Es así que no consideramos los siguientes code smells, al no utilizar un paradigma orientado a objetos :

- **God Class** .- Clase larga y compleja que centraliza la inteligencia del sistema.
  - **Data class** .- Clase que contiene datos pero no comportamiento relacionado con los datos
  - **Feature envy** .- Método que llama a más métodos de una sola clase externa que los métodos internos de su propia clase interna.
  - **Refused bequest** .- Subclase que no usa los métodos protegidos de su superclase.
  - Tradition breaker .- Subclase que proporciona un gran conjunto de servicios que no están relacionados con los servicios proporcionados por la superclase.
- 
- **Nombres inapropiados.**- Ciertas archivos o variables podrían estar mal nombradas, no haciendo referencia a su propósito o no brindando la ayuda para entender el código.



- **Comentarios.-** Para el entendimiento del código es necesario tener comentado que hace cada parte del código, así si en algún momento es necesario que alguien externo al manejo del código necesita realizar alguna modificación o mejora se le haga fácil.

```
import express from 'express'
import config from './config'
import estudiantesRoutes from './routes/Estudiantes.routes'
import docentesRoutes from './routes/Docentes.routes'
const cors=require('cors');
//usamos el framework express para la creacion del servidor
const app=express();
//Cors para la comunicacion entre front y back
app.use(cors());
//settings
//definir el puerto dentro de app
app.set('port',config.port);

//middlewares
app.use(express.json());//para poder recibir json desde el cliente
app.use(express.urlencoded({extended:false}));//para poder recibir

//port
//usamos todas las rutas de la api para estudiantes,docente
app.use(estudiantesRoutes);
app.use(docentesRoutes);
```

- **Código muerto.-** Cuando los requisitos del software han cambiado o se han realizado correcciones, nadie ha tenido tiempo de limpiar el código antiguo.

## Código no usado

```
src > Administracion > AdminDocentes.jsx > AdminDocentes > useEffect() callback
132     promise.then(() => {
133         setExcel(d);
134     })
135 }
136 const abrirCerrarModalInsertar={()=>{
137     setModalInsertar(!modalInsertar);
138 }
139 const abrirCerrarModalWarning={()=>{
140     setWarningview(!warningView);
141 }
142 //const guardarExcel={()=>{
143 //    console.log(excel)
144 //    document.getElementById('inputGroupFile04').value = '';
145 //}
146 useEffect(()=>{
147     peticionGet();
148 })
```

## Componente: Hacer Tutor

```
import React from 'react'
import AdminBar from '../Administracion/AdminBar'
import {Col,Row} from 'react-bootstrap'
import '../styles/AdminHacerTutor.css'
const AdminHacerTutor = () => {
  return (
    <div>
      <AdminBar/>
      <div className="contenido">
        <div className="Principal2">
          <div className="cont">
            <h5>Lista de docentes :</h5>
            <div className="TablaHacerTutor">
              <div className="col tableScrollHacerTutor scrollHacerTutor">
                <table className="table table-bordered bg-light ">
                  <thead>
                    <tr>
                      <th>Nro</th>
                      <th>Curso</th>
                      <th>Estudiante</th>
                      <th>Detalles</th>
                    </tr>
                  </thead>
                  <tbody>
                    <tr>
```

- **Código duplicado.-** El código duplicado es un código que se repite en diferentes lugares haciendo lo mismo. Los fragmentos de código que son muy similares también pueden considerarse duplicados. En general las consultas que se realizan son muy similares para los CRUD de docente y estudiante, haciendo un code smell.

## El método agregar estudiante

```
src > controllers > Estudiantes.controllers.js > getEstudianteById
48 export const addEstudiantes=async (req,res)=>{
49   try{
50     const Lista=req.body;
51     let quieriestemp='';
52     for (let i = 0; i < Lista.length; i++) {
53       let CodEstudiante=Lista[i].CodEstudiante,Nombres=Lista[i].Nombres,
54       Email=Lista[i].Email,Direccion=Lista[i].Direccion,Celular=Lista[i].Celular;
55       quieriestemp+="Insert into TEstudiante Values ('"+CodEstudiante+"',
56     }
57     console.log(quieriestemp);
58     const pool=await getConnection();
59     const result=await pool.request().query(quieriestemp);
60     console.log('addEstudiantes executed')
61     res.json(result.recordset);
62   }catch(error){
63     res.status(500);
64     res.send(error.message);}
65 };
```

## El método agregar docentes

```
src > controllers > JS Docentes.controllers.js > addDocentes > CodDocente
62 export const addDocentes=async (req,res)=>{
63   try{
64     const Lista=req.body;
65     let quieriestemp='';
66     for (let i = 0; i < Lista.length; i++) {
67       let CodDocente=Lista[i].CodDocente,Nombres=Lista[i].Nombres,ApPater
68       DNI=Lista[i].DNI,Categoria=Lista[i].Categoria,Celular=Lista[i].Celu
69       EsTutor=Lista[i].EsTutor;
70       quieriestemp+="Insert into TDocente Values ('"+CodDocente+"','"+Nomb
71     }
72     console.log(quieriestemp);
73     const pool=await getConnection();
74     const result=await pool.request().query(quieriestemp);
75     console.log('addDocentes executed')
76     res.json(result.recordset);
77   }catch(error){
78     res.status(500);
79     res.send(error.message);}
80 };
```

- **Código largo.-** Los programadores generalmente encuentran mentalmente menos agotador colocar una nueva característica en una clase existente que crear una nueva clase para la característica.

```
src > Administracion > AdminDocentes.jsx > AdminDocentes
64 const peticionGet=async()=>{
65   await axios.get(baseUrl)
66   .then(response=>{
67     setData(response.data);
68   })
69   .catch(error=>{
70     console.log(error);
71   })
72 }
73 const peticionPostExcel=async()=>{
74   await axios.post(baseUrlExcel,excel)
75   .then(response=>{
76     setData(data.concat(response.data));
77     limpiar();
78     document.getElementById('inputGroupFile04').value = '';
79   }).catch(error=>{
80     console.log(error);
81   })
82 }
83 const peticionPost=async()=>{
84   if(!codDocente.trim()||!nombres.trim()||!dni.trim()||!apPaterno.trim()||!ap
85     setWarningview(true)
86     return
87 }
```



```

tracion > AdminDocentes.jsx > AdminDocentes
return (
  <div>
    <AdminBar nombrePage={"Docentes"}/>
    <div className="contenido">
      <div className="Principal2">
        <div className="cont">
          <h5>Lista de docentes:</h5>
          <div className="TablaDT">
            <div className="col tableScrollDT scrollDT">
              <table className="table table-bordered bg-light ">
                <thead className="colTable">
                  <tr>
                    <th>CodDocente</th>
                    <th>DNI</th>
                    <th>Nombres</th>
                    <th>Apellidos</th>
                    <th>Categoria</th>
                    <th>Celular</th>
                    <th>Email</th>
                    <th>Direccion</th>
                  </tr>
                </thead>

```

- **Deep Indentation.-** Se evita la legibilidad del código.

```

src > Administracion > AdminDocentes.jsx > AdminDocentes > readExcel > promise > <function>
113
114   const readExcel=(file)=>{
115     const promise=new Promise((resolve,reject)=>{
116       const fileReader=new FileReader();
117       fileReader.readAsArrayBuffer([file])
118       fileReader.onload=(e)=>{
119         const bufferArray=e.target.result;
120         const wb=XSX.read(bufferArray,{type:'buffer'});
121
122         const wsname=wb.SheetNames[0];
123
124         const ws=wb.Sheets[wsname];
125         const data=XSX.utils.sheet_to_json(ws);
126         resolve(data);
127       };
128       fileReader.onerror=((error)=>{
129         reject(error);
130       })
131     })

```

- **Uso de muchos useState():**

Un componente con muchos useState() hooks probablemente esté haciendo demasiadas cosas y probablemente sea un buen candidato para dividirse en múltiples componentes, pero también hay algunos casos complejos en los que necesitamos administrar algún estado complejo en un solo componente.

```
});  
const[modalInsertar,setModalInsertar]=useState(false);  
const[modalActualizar,setModalActualizar]=useState(false);  
const[codDocente,setCodDocente]=useState('')  
const[nombres,setNombres]=useState('')  
const[dni,setDni]=useState('')  
const[apPaterno,setApPaterno]=useState('')  
const[apMaterno,setApMaterno]=useState('')  
const[categoria,setCategoria]=useState('')  
const[celular,setCelular]=useState('')  
const[email,setEmail]=useState('')  
const[direccion,setDireccion]=useState('')  
const[esTutor,setEsTutor]=useState('')  
const[warningView,setWarningview]=useState(false);  
/*metodos para el ani*
```

**Deuda Técnica:** Trabajo que se adquiere al producir código pobre, incumpliendo prácticas aconsejadas para el desarrollo de software

- **Documentación escasa, incompleta o inservible**

En el proceso de la formación del nuevo grupo, se junto la documentación planteada por ambos grupos y se realizó un estudio evaluando qué criterios tomar para continuar el proyecto, esto nos llevó a tener documentación de más que al final se tuvo que desechar, así mismo se hizo la adecuación de requisitos para su cumplimiento y la actualización de ciertos archivos.

- **Arquitectura no escalable**

El diseño del sistema que estamos desarrollando no cuenta con una arquitectura preestablecida para el desarrollo en general, es así que no consideramos que permita hacer el mantenimiento adecuado a los componentes de manera adecuada, ralentizando la inclusión futura de nuevas funcionalidades al software.

- **Ausencia o deficiente control de versiones.**

- **Rigidez para actualizar a nuevas tecnologías o plataformas.**

**¿Por qué hemos caído en estos Code smells?**

- Falta de experiencia en la utilización de frameworks de desarrollo web.
- Malos hábitos de programación
- Inexistencia de una arquitectura de software específica

## **Referencias**

- <https://medium.com/oceanize-geeks/code-smells-and-refactoring-c2coe0642582>
- <https://refactoring.guru/es/refactoring/smells>
- <https://antongunnarsson.com/react-component-code-smells/>
- <https://hackernoon.com/lessons-learned-common-react-code-smells-and-how-to-avoid-them-f253eb9696a4>
- <https://betterprogramming.pub/looking-for-code-smells-in-javascript-677f1a312f29>

# Patrones de Diseño

## PATRONES DE DISEÑO

### ¿Qué son los patrones de diseño?

Los patrones de diseño o design patterns, son una solución general, reutilizable y aplicable a diferentes problemas de diseño de software. Tiene como objetivo identificar problemas en el sistema y proporcionar soluciones apropiadas a problemas generales a los que se han enfrentado los desarrolladores durante un largo periodo de tiempo, a través de prueba y error.

### Tipos de patrones de diseño de software



# PATRONES DE DISEÑO EN BACKEND

## 1. Patrones creacionales

- a. **Singleton** es un patrón de diseño creacional que nos permite asegurarnos de que una clase tenga una única instancia, a la vez que proporciona un punto de acceso global a dicha instancia.

En el manejo del backend tomamos la instancia al iniciar la conexión de manera global, y de esta manera poder consumirla mediante diferentes rutas que están destinadas para cierto propósito como lo son la asignación de tutores o el manejo de los tutores.

```
src > app.js > ...
1  import express from 'express'
2  import config from './config'
3  import estudiantesRoutes from './routes/Estudiantes.routes'
4  import docentesRoutes from './routes/Docentes.routes'
5  import sesionesRoutes from './routes/SesionTutoria.routes'
6  import asignacionesRoutes from './routes/Asignaciones.routes'
7  import others from './routes/Others.routes'
8  import fichasRoutes from './routes/FichasTutoria.routes'
9  const cors=require('cors');
10 //usamos el framework express para la creacion del servidor
11 const app=express();
12 //Cors para la comunicacion entre front y back
13 app.use(cors());
14 //settings
15 //definir el puerto dentro de app
16 app.set('port',config.port);
17 //middlewares
18 app.use(express.json());//para poder recibir json desde el cliente
19 app.use(express.urlencoded({extended:false}));//para poder recibir datos d
20 //port
21 //usamos todas las rutas de la api para estudiantes,docente
22 app.use(estudiantesRoutes);
23 app.use(docentesRoutes);
24 app.use(asignacionesRoutes);
25 app.use(sesionesRoutes);
26 app.use(fichasRoutes)
27 app.use(others);
28
29 export default app;
```

app.js

## 2. Patrones estructurales

- a. Bridge** es un patrón de diseño estructural que te permite dividir una clase grande, o un grupo de clases estrechamente relacionadas, en dos jerarquías separadas (abstracción e implementación) que pueden desarrollarse independientemente la una de la otra.

Esto lo podemos ver a nivel de backend al momento de disgregar ciertos archivos como `Docentes.routes.js` que necesita y llama módulos de `Docentes.controllers.js` y este también a su vez llama a las consultas del archivo `qurys.js`

```

src > routes > Docentes.routes.js > default
1  import { Router } from "express";
2
3  import {addDocente, addDocentes,getTutorById, deleteDocenteById,
4      getDocenteById, getDocentes, updateDocenteById,getTutores,
5      loginDocente, loginCoordinador,getCoordinador}
6      from '../controllers/Docentes.crontrollers'
7  //DOCENTES ROUTES
8  //importamos la funcion router para el enrutado
9  const router=Router();
10 //funcion para obtener todos los docente
11 router.get('/docentes',getDocentes);
12 //funcion para obtener una docente por id
13 router.get('/docentes/:id',getDocenteById);
14 //funcion para agregar un docente nuevo
15 router.post('/docentes',addDocente);
16 //funcion para agregar varios estudiantes
17 router.post('/docentesLista',addDocentes);
18 //funcion para actualizar un docente por ID
19 router.put('/docentes/:id',updateDocenteById);
20 //funcion para eliminar un docente por ID
21 //riesgo si se elimina estudiante se deben implementar funciones
22 //en cascada para eliminar correctamente
23 router.delete('/docen: Docentes.controllers.js
24 //funcion para recuperar src > controllers > Docentes.controllers.js > getDoc

```

Docentes.routes.js

Docentes.controls.js

```
src > controllers > Docentes.controllers.js > [0] getDocenteById
1 import { getConnection,sql,queries } from "../database";
2 //peticiones a la base de datos se detalla la funcionalidad en Estudiantes.routes.js
3 export const getDocentes=async (req,res)=>{
4     try{
5         const pool=await getConnection()
6         const result=await pool.request().query(queries.getAllDocentes);
7         console.log("getDocentes executed");
8         res.json(result.recordset)
9     }catch(error){
10         res.status(500);
11         res.send(error.message);
12     }
13 };
14 export const getDocenteById=async (req,res)=>{
15     try{
16         const { id }=req.params;
17         const pool=await getConnection();
18         const result=await pool.request().query(queries.getDocenteById(id));
19         console.log("getDocenteById executed");
20         res.json(result.recordset)
21     }catch(error){
22         res.status(500);
23         res.send(error.message);
24     }
25 };
26 export const getDocenteByCedula=async (req,res)=>{
27     try{
28         const { cedula }=req.params;
29         const pool=await getConnection();
30         const result=await pool.request().query(queries.getDocenteByCedula(cedula));
31         console.log("getDocenteByCedula executed");
32         res.json(result.recordset)
33     }catch(error){
34         res.status(500);
35         res.send(error.message);
36     }
37 };
38 export const getDocenteByNombre=async (req,res)=>{
39     try{
40         const { nombre }=req.params;
41         const pool=await getConnection();
42         const result=await pool.request().query(queries.getDocenteByNombre(nombre));
43         console.log("getDocenteByNombre executed");
44         res.json(result.recordset)
45     }catch(error){
46         res.status(500);
47         res.send(error.message);
48     }
49 };
50 export const getDocenteByApellido=async (req,res)=>{
51     try{
52         const { apellido }=req.params;
53         const pool=await getConnection();
54         const result=await pool.request().query(queries.getDocenteByApellido(apellido));
55         console.log("getDocenteByApellido executed");
56         res.json(result.recordset)
57     }catch(error){
58         res.status(500);
59         res.send(error.message);
60     }
61 };
62 export const getDocenteByCedulaYApellido=async (req,res)=>{
63     try{
64         const { cedula, apellido }=req.params;
65         const pool=await getConnection();
66         const result=await pool.request().query(queries.getDocenteByCedulaYApellido(cedula,apellido));
67         console.log("getDocenteByCedulaYApellido executed");
68         res.json(result.recordset)
69     }catch(error){
70         res.status(500);
71         res.send(error.message);
72     }
73 };
74 export const getDocenteByCedulaYNombre=async (req,res)=>{
75     try{
76         const { cedula, nombre }=req.params;
77         const pool=await getConnection();
78         const result=await pool.request().query(queries.getDocenteByCedulaYNombre(cedula,nombre));
79         console.log("getDocenteByCedulaYNombre executed");
80         res.json(result.recordset)
81     }catch(error){
82         res.status(500);
83         res.send(error.message);
84     }
85 };
86 export const getDocenteByCedulaYNombreYApellido=async (req,res)=>{
87     try{
88         const { cedula, nombre, apellido }=req.params;
89         const pool=await getConnection();
90         const result=await pool.request().query(queries.getDocenteByCedulaYNombreYApellido(cedula,nombre,apellido));
91         console.log("getDocenteByCedulaYNombreYApellido executed");
92         res.json(result.recordset)
93     }catch(error){
94         res.status(500);
95         res.send(error.message);
96     }
97 };
98 export const getDocenteByCedulaYNombreYApellidoYCedula=async (req,res)=>{
99     try{
100         const { cedula, nombre, apellido, cedula2 }=req.params;
101         const pool=await getConnection();
102         const result=await pool.request().query(queries.getDocenteByCedulaYNombreYApellidoYCedula(cedula,nombre,apellido,cedula2));
103         console.log("getDocenteByCedulaYNombreYApellidoYCedula executed");
104         res.json(result.recordset)
105     }catch(error){
106         res.status(500);
107         res.send(error.message);
108     }
109 };
110 export const getDocenteByCedulaYNombreYApellidoYCedulaYNombre=async (req,res)=>{
111     try{
112         const { cedula, nombre, apellido, cedula2, nombre2 }=req.params;
113         const pool=await getConnection();
114         const result=await pool.request().query(queries.getDocenteByCedulaYNombreYApellidoYCedulaYNombre(cedula,nombre,apellido,cedula2,nombre2));
115         console.log("getDocenteByCedulaYNombreYApellidoYCedulaYNombre executed");
116         res.json(result.recordset)
117     }catch(error){
118         res.status(500);
119         res.send(error.message);
120     }
121 };
122 export const getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellido=async (req,res)=>{
123     try{
124         const { cedula, nombre, apellido, cedula2, nombre2, apellido2 }=req.params;
125         const pool=await getConnection();
126         const result=await pool.request().query(queries.getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellido(cedula,nombre,apellido,cedula2,nombre2,apellido2));
127         console.log("getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellido executed");
128         res.json(result.recordset)
129     }catch(error){
130         res.status(500);
131         res.send(error.message);
132     }
133 };
134 export const getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedula=async (req,res)=>{
135     try{
136         const { cedula, nombre, apellido, cedula2, nombre2, apellido2, cedula3 }=req.params;
137         const pool=await getConnection();
138         const result=await pool.request().query(queries.getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedula(cedula,nombre,apellido,cedula2,nombre2,apellido2,cedula3));
139         console.log("getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedula executed");
140         res.json(result.recordset)
141     }catch(error){
142         res.status(500);
143         res.send(error.message);
144     }
145 };
146 export const getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombre=async (req,res)=>{
147     try{
148         const { cedula, nombre, apellido, cedula2, nombre2, apellido2, cedula3, nombre3 }=req.params;
149         const pool=await getConnection();
150         const result=await pool.request().query(queries.getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombre(cedula,nombre,apellido,cedula2,nombre2,apellido2,cedula3,nombre3));
151         console.log("getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombre executed");
152         res.json(result.recordset)
153     }catch(error){
154         res.status(500);
155         res.send(error.message);
156     }
157 };
158 export const getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellido=async (req,res)=>{
159     try{
160         const { cedula, nombre, apellido, cedula2, nombre2, apellido2, cedula3, nombre3, apellido3 }=req.params;
161         const pool=await getConnection();
162         const result=await pool.request().query(queries.getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellido(cedula,nombre,apellido,cedula2,nombre2,apellido2,cedula3,nombre3,apellido3));
163         console.log("getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellido executed");
164         res.json(result.recordset)
165     }catch(error){
166         res.status(500);
167         res.send(error.message);
168     }
169 };
170 export const getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedula=async (req,res)=>{
171     try{
172         const { cedula, nombre, apellido, cedula2, nombre2, apellido2, cedula3, nombre3, apellido3, cedula4 }=req.params;
173         const pool=await getConnection();
174         const result=await pool.request().query(queries.getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedula(cedula,nombre,apellido,cedula2,nombre2,apellido2,cedula3,nombre3,apellido3,cedula4));
175         console.log("getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedula executed");
176         res.json(result.recordset)
177     }catch(error){
178         res.status(500);
179         res.send(error.message);
180     }
181 };
182 export const getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombre=async (req,res)=>{
183     try{
184         const { cedula, nombre, apellido, cedula2, nombre2, apellido2, cedula3, nombre3, apellido3, cedula4, nombre4 }=req.params;
185         const pool=await getConnection();
186         const result=await pool.request().query(queries.getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombre(cedula,nombre,apellido,cedula2,nombre2,apellido2,cedula3,nombre3,apellido3,cedula4,nombre4));
187         console.log("getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombre executed");
188         res.json(result.recordset)
189     }catch(error){
190         res.status(500);
191         res.send(error.message);
192     }
193 };
194 export const getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellido=async (req,res)=>{
195     try{
196         const { cedula, nombre, apellido, cedula2, nombre2, apellido2, cedula3, nombre3, apellido3, cedula4, nombre4, apellido4 }=req.params;
197         const pool=await getConnection();
198         const result=await pool.request().query(queries.getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellido(cedula,nombre,apellido,cedula2,nombre2,apellido2,cedula3,nombre3,apellido3,cedula4,nombre4,apellido4));
199         console.log("getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellido executed");
200         res.json(result.recordset)
201     }catch(error){
202         res.status(500);
203         res.send(error.message);
204     }
205 };
206 export const getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedula=async (req,res)=>{
207     try{
208         const { cedula, nombre, apellido, cedula2, nombre2, apellido2, cedula3, nombre3, apellido3, cedula4, nombre4, apellido4, cedula5 }=req.params;
209         const pool=await getConnection();
210         const result=await pool.request().query(queries.getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedula(cedula,nombre,apellido,cedula2,nombre2,apellido2,cedula3,nombre3,apellido3,cedula4,nombre4,apellido4,cedula5));
211         console.log("getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedula executed");
212         res.json(result.recordset)
213     }catch(error){
214         res.status(500);
215         res.send(error.message);
216     }
217 };
218 export const getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombre=async (req,res)=>{
219     try{
220         const { cedula, nombre, apellido, cedula2, nombre2, apellido2, cedula3, nombre3, apellido3, cedula4, nombre4, apellido4, cedula5, nombre5 }=req.params;
221         const pool=await getConnection();
222         const result=await pool.request().query(queries.getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombre(cedula,nombre,apellido,cedula2,nombre2,apellido2,cedula3,nombre3,apellido3,cedula4,nombre4,apellido4,cedula5,nombre5));
223         console.log("getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombre executed");
224         res.json(result.recordset)
225     }catch(error){
226         res.status(500);
227         res.send(error.message);
228     }
229 };
230 export const getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellido=async (req,res)=>{
231     try{
232         const { cedula, nombre, apellido, cedula2, nombre2, apellido2, cedula3, nombre3, apellido3, cedula4, nombre4, apellido4, cedula5, nombre5, apellido5 }=req.params;
233         const pool=await getConnection();
234         const result=await pool.request().query(queries.getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellido(cedula,nombre,apellido,cedula2,nombre2,apellido2,cedula3,nombre3,apellido3,cedula4,nombre4,apellido4,cedula5,nombre5,apellido5));
235         console.log("getDocenteByCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellidoYCedulaYNombreYApellido executed");
236         res.json(result.recordset)
237    
```

```

ys.js
database > queries.js > queries
export const queries=[
  // Queries Docentes
  getAllDocentes:"Select * from TDocente",
  getDocenteById:"Select * from TDocente where CodDocente=@CodDocente",
  addNewDocente:"Insert into TDocente Values (@CodDocente,@Nombres,@ApPaterno,@ApMaterno,@DNI,@
  deleteDocenteById:"delete from TDocente where CodDocente=@CodDocente",
  updateDocenteById:"update TDocente set Categoria=@Categoria, Email=@Email,Celular=@Celular,Di
  getTutores:"Select * from TDocente where esTutor='Si'",
  loginDocente:"execute spuVerificacionLoginDocente @Usuario,@Contrasenia;",
  loginCoordinador:"execute spuVerificacionLoginCoordinador @Usuario,@Contrasenia;",
  getTutorById:"Select * from TDocente where (CodDocente=@CodDocente and esTutor='Si')".

```

- b. Decorator** es un patrón de diseño estructural que te permite añadir funcionalidades a objetos colocando estos objetos dentro de objetos encapsuladores especiales que contienen estas funcionalidades.



Este patrón se puede visualizar en los archivos controller donde se pueden funcionalidades como métodos get, set o put que podrían ser necesarios o requeridos, sin tener que modificar la distribución de los archivos base.

```
FichasTutoria.controllers.js X
src > controllers > FichasTutoria.controllers.js > ...
1 import { getConnection,sql,queries } from "../database";
2 //peticiones a la base de datos se detalla la funcionalidad en FichasTutoria.routes.js
3 export const getFichas=async (req,res)=>{
4   try{
5     const pool=await getConnection()
6     const result=await pool.request().query(queries.getAllFichas);
7     console.log('getFichas executed');
8     res.json(result.recordset)
9   }catch(error){
10     res.status(500);
11     res.send(error.message);
12   }
13 };
14 export const getFichaById=async (req,res)=>{
15   try{
16     const { id }=req.params;
17     const pool=await getConnection();
18     const result=await pool.request().input("IdFichaTutoria",sql.VarChar,id)
19     .query(queries.getFichaById);
20     console.log('getFichaById executed',id);
21     res.json(result.recordset);
22   }catch(error){
23     res.status(500);
24     res.send(error.message);
25   }
26 };
27 export const addFicha=async (req,res)=>{
28   try{
29     const {IdFichaTutoria,IdAsignacion,CelularReferenciaTutorando,PersonaReferenciaTutorando} = req.body;
30     const pool=await getConnection();
31     await pool.request()
32       .input("IdFichaTutoria",sql.VarChar,IdFichaTutoria)
33       .input("IdAsignacion",sql.VarChar,IdAsignacion)
34       .input("CelularReferenciaTutorando",sql.VarChar,CelularReferenciaTutorando)
35       .input("PersonaReferenciaTutorando",sql.VarChar,PersonaReferenciaTutorando)
36       .query(queries.addNewFicha);
37     console.log('addFicha executed',IdFichaTutoria)
38     res.json({IdFichaTutoria});
39   }catch(error){
40     res.status(500);
41     res.send(error.message);
42   }
43 }
```


FichasTutoria.controllers.js

### 3. Patrones de comportamiento en node.js

- a. **Chain of responsibility** consiste en estructurar tu código de manera que te permita desvincular de una solicitud, estamos creando una cadena de peticiones de recepción, que intentarán cumplir con la solicitud y, si no pueden, simplemente la pasarán. Esto se puede ver al validar la configuración de la conexión con la base de datos, donde si por error algún dato de la configuración está mal, se usará una configuración por defecto establecida.

```
rc >  config.js >  default
1  import { config } from "dotenv"
2  config();
3  //configuracion de los parametros en variables de entorno
4  export default{
5      port:process.env.PORT || 5000,
6      dbUser:process.env.DB_USER || '',
7      dbPassword:process.env.DB_PASSWORD || '',
8      dbServer:process.env.DB_SERVER || '',
9      dbDatabase:process.env.DB_DATABASE || ''
10 };
```

config.js

```
 .env
1  NICKNAME = DAN
2  PORT = 4000
3  DB_USER= adminBD
4  DB_PASSWORD= 1-password2
5  DB_SERVER= sqltutoria-server.database.windows.net
6  DB_DATABASE= BDSistema_Tutorias
7
```

.env



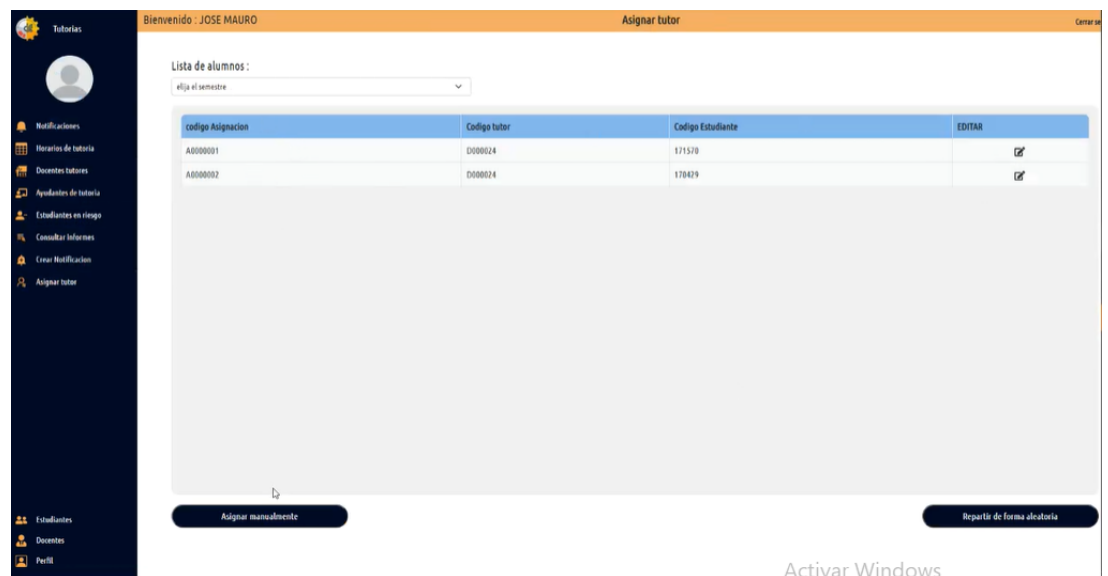
# PATRONES DE DISEÑO EN FRONTEND

## 1. Patrones creacionales

- a. **Factory Method:** Se desarrolla este patrón para la creación de elementos de interfaz de usuario, en las vistas de la interfaz de tutor, tutorado y administración podemos visualizar cómo es que hacemos uso de este patrón, notando que existe en un `className="contenido"` que hace la función de un contenedor donde será llamado constantemente en cada vista, con la diferencia de que en cada caso, los objetos que va a contener serán de acuerdo a las necesidades de la interfaz.

```
<TutoradoBar/>
<div className="contenido">
  <div className=" Principal2">
    <AdminBar nombrePage={"Estudiantes"} />
    <div className="contenido">
      <div className="Principal2">
```

```
<Tutorbar/>
<div className="contenido">
  <div className="Principal2">
```



- b. **Builder:** Este patrón de diseño nos permite construir objetos complejos paso a paso, en el desarrollo de la vista perfil del Tutor hacemos uso del patrón, creando objetos necesarios para lograr el objetivo y lo mismo sucede para la vista del Tutorado.

```


<div className="contDatos">
  <label className="lbldatos" htmlFor=""><b>Datos Personales :</b></label>
  <Row className="position-relative">
    <Col className="column1">
      <div>
        <label htmlFor=""><b>Nombres :</b></label>
        <label className="lbldat"> {cookie.get('Nombres')}</label>
      </div>
      <div>
        <label htmlFor=""><b>Apellidos :</b></label>
        <label className="lbldat"> {cookie.get('ApPaterno')}+" "+cookie.get('ApMaterno')</label>
      </div>
      <div>
        <label htmlFor=""><b>Email :</b></label>
        <label className="lbldat"> {cookie.get('Email')}</label>
      </div>
    </Col>
    <Col className="column1">
      <div>
        <label htmlFor=""><b>Direccion :</b></label>
        <label className="lbldat"> {cookie.get('Direccion')}</label>
      </div>
    </Col>
  </Row>
</div>
```

- c. **Singleton:** Este patrón de diseño creacional nos permite asegurarnos de que una clase tenga una única instancia, en el desarrollo del Frontend se puede visualizar este patrón para tener el control de las instancias de las rutas de las interfaces, llamar o crear el objeto que devuelve cada ruta.

```
import LoginMenu from './components/log_menu';
import LoginTutorados from './Tutorados/LoginTutorados';
import LoginTutor from './Tutor/LoginTutor';
import LoginAdministracion from './Administracion/LoginAdmin';
import TadoMenu from './Tutorados/TadoMenu';
import Prueba from './components/prueba';
import TadoHorarios from './Tutorados/TadoHorarios';
import {Switch, Route, BrowserRouter} from "react-router-dom";
import TadoTutorAsignado from './Tutorados/TadoTutorAsignado';
import TadoEstudianteAsignado from './Tutorados/TadoEstudianteAsignado';
import TadoPerfil from './Tutorados/TadoPerfil';
import TutorMenu from './Tutor/TutorMenu';
import TutorEstudiantesCargo from './Tutor/TutorEstudiantesCargo';
import TutorEstudiantesAyudantes from './Tutor/TutorEstudiantesAyudantes';
import TutorInformeQuincenal from './Tutor/TutorInformeQuincenal';
import TutorInformeSemestral from './Tutor/TutorInformeSemestral';
import TutorObtencionNotas from './Tutor/TutorObtencionNotas';
import TutorRegistrarFichaTutoria from './Tutor/TutorRegistrarFichaTutoria';
import TutorPerfil from './Tutor/TutorPerfil';
import AdminMenu from './Administracion/AdminMenu';
import AdminHorarios from './Administracion/AdminHorarios';
import AdminDocentesTutores from './Administracion/AdminDocentesTutores';
import AdminAyudantes from './Administracion/AdminAyudantes';
import AdminEstudiantesRiesgo from './Administracion/AdminEstudiantesRiesgo';
import AdminConsultarInfo from './Administracion/AdminConsultarInfo';
import AdminCrearNotifi from './Administracion/AdminCrearNotifi';
import AdminAsignarTutor from './Administracion/AdminAsignarTutor';
import AdminEstudiantes from './Administracion/AdminEstudiantes';
import AdminDocentes from './Administracion/AdminDocentes';
import AdminPerfil from './Administracion/AdminPerfil';
```

```
<Route exact path="/LogMenu" component={LoginMenu}/>
<Route exact path="/LoginTutorados" component={LoginTutorados}/>
<Route exact path="/LoginTutor" component={LoginTutor}/>
<Route exact path="/LoginAdministracion" component={LoginAdministracion}/>
<Route exact path="/Tutorado_Menu" component={TadoMenu}/>
<Route exact path="/Tutorado_Horarios" component={TadoHorarios}/>
<Route exact path="/Tutorado_TutorAsignado" component={TadoTutorAsignado}/>
<Route exact path="/Tutorado_EstudianteAsignado" component={TadoEstudianteAsignado}/>
<Route exact path="/Tutorado_Perfil" component={TadoPerfil}/>
<Route exact path="/Tutor_Menu" component={TutorMenu}/>
<Route exact path="/Tutor_Estudiantes_a_cargo" component={TutorEstudiantesCargo}/>
<Route exact path="/Tutor_Estudiantes_Ayudantes" component={TutorEstudiantesAyudantes}/>
<Route exact path="/Tutor_Informe_Quincenal" component={TutorInformeQuincenal}/>
<Route exact path="/Tutor_Informe_Semestral" component={TutorInformeSemestral}/>
<Route exact path="/Tutor_Registrar_Ficha_Tutoria" component={TutorRegistrarFichaTutoria}/>
<Route exact path="/Tutor_Sesion_Tutoria" component={TutorSesionTutorias}/>
<Route exact path="/Tutor_Obtencion_Notas" component={TutorObtencionNotas}/>
<Route exact path="/Tutor_Perfil" component={TutorPerfil}/>
<Route exact path="/Admin_Menu" component={AdminMenu}/>
<Route exact path="/Admin_Horarios" component={AdminHorarios}/>
<Route exact path="/Admin_Docentes_Tutores" component={AdminDocentesTutores}/>
<Route exact path="/Admin_Ayudantes" component={AdminAyudantes}/>
<Route exact path="/Admin_Estudiantes_Riesgo" component={AdminEstudiantesRiesgo}/>
<Route exact path="/Admin_Consultar_Informes" component={AdminConsultarInfo}/>
<Route exact path="/Admin_Crear_Notificacion" component={AdminCrearNotifi}/>
<Route exact path="/Admin_Asignar_Tutor" component={AdminAsignarTutor}/>
<Route exact path="/Admin_Estudiantes" component={AdminEstudiantes}/>
<Route exact path="/Admin_Docentes" component={AdminDocentes}/>
<Route exact path="/Admin_Perfil" component={AdminPerfil}/>
```

## 2. Patrones estructurales

- a. **Adapter:** Este patrón de diseño estructural permite la colaboración entre objetos con interfaces compatibles.

Desarrollamos un adaptador para cuando se suba datos en un archivo de excel estos ingresen a la biblioteca de análisis a través de los métodos adecuados, además que existe la compatibilidad entre la interfaz y el objeto.

```
const petitionPostExcel=async()=>{
  await axios.post(baseUrlExcel,excel)
  .then(response=>{
    setData(data.concat(response.data));
    limpiar();
    document.getElementById('inputGroupFile04').value = '';
  }).catch(error=>{
    console.log(error);
  })
}
```

- b. **Composite:** Este patrón de diseño estructural permite componer objetos estructurales y trabajar con estas como si fueran individuales, para el desarrollo del frontend podemos encontrar gran cantidad de este patrón ya que dentro de container llamamos a un modal y este también contiene otros objetos como labels, entradas y botones.

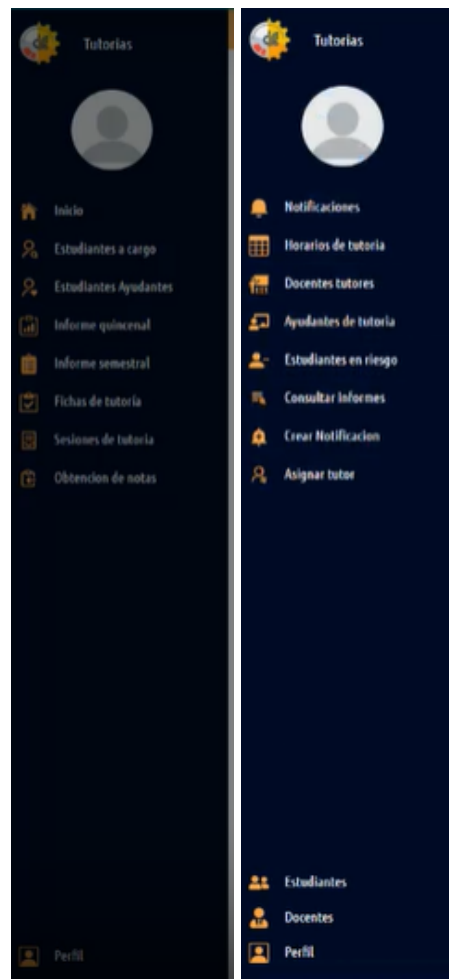
```
<ModalHeader>Asignar</ModalHeader>

<ModalBody>
  <Col>
    <Row>
      <Col className="col-4">
        <h6>Codigo Estudiante :</h6>
      </Col>
      <Col className="col-8">
        <select value={codEstudiante} onChange={(e) => {setCodEstudiante(e.target.value)}}className="form-select form-select-sm">
          <option value="codigo estudiante">codigo estudiante</option>
          {
            estudiantes.map((item, index) => (
              <option key={index} value={item.CodEstudiante}>{item.CodEstudiante}----{item.Nombres} {item.ApPaterno},{item.ApMaterno} </option>
            ))
          }
        </select>
      </Col>
    </Row>
    <hr />
    <Row>
      <Col className="col-4">
        <h6>Codigo Docente :</h6>
      </Col>
      <Col className="col-8">
        <select value={codDocente} onChange={(e) => {setCodDocente(e.target.value)}}className="form-select form-select-sm">
          <option value="codigo docente">codigo docente</option>
          {
            tutoresList.map((item, index) => (
              <option key={index} value={item.CodDocente}>{item.CodDocente}----{item.Nombres} {item.ApPaterno},{item.ApMaterno} </option>
            ))
          }
        </select>
      </Col>
    </Row>
  </ModalBody>
</Modal>
```

- c. **Decorator:** Este patrón de diseño estructural que nos permite añadir funcionalidades a los objetos colocando estos objetos dentro de otros, encapsulando los. Composite y Decorator tienen diagramas de estructura similares ya que ambos se basan en la composición recursiva para organizar un número indefinido de objetos

### 3. Patrones de comportamiento

- a. **State** es un patrón de diseño estructural que te permite añadir funcionalidades a objetos colocando estos objetos dentro de objetos encapsuladores especiales que contienen estas funcionalidades. Es un patrón de diseño de comportamiento que permite a un objeto alterar su comportamiento cuando su estado interno cambia.



- b. **Chain of responsibility:** Es un patrón de diseño de comportamiento que te permite pasar solicitudes a lo largo de una cadena de manejadores. Al

recibir una solicitud, cada manejador decide si la procesa o si la pasa al siguiente manejador de la cadena.

En el front end podemos observar que en la vista del login se realiza una validación para que pueda pasar a la siguiente vista ya sea tutor, turado o administración según corresponda



## Referencias

- <https://blog.risingstack.com/fundamental-node-js-design-patterns/>
- <https://blog.logrocket.com/design-patterns-in-node-js/>
- <https://refactoring.guru/es/design-patterns>
- <https://springframework.guru/gang-of-four-design-patterns/>
- <https://refactoring.guru/es/design-patterns/chain-of-responsibility>

# **Métricas de Software**

## **1. Análisis Funcional(Función de puntos de análisis)**

### **a. Base Teórica**

Está definido como un método para medir el desarrollo de software desde el punto de vista del usuario. En su funcionamiento, mediante la asignación de “puntos” identifica los componentes del sistema en términos de transacciones y grupos de datos lógicos que son relevantes para el usuario en su negocio.

Los puntos de función miden el tamaño de una aplicación planificada (lógico) o existente (funcional), también puede ser usado para medir el tamaño de los cambios de una aplicación existente.

El proceso sería el siguiente:

- o. Obtener información del sistema
1. Identificar componentes del sistema
2. Calcular el número de elementos y su complejidad
3. Obtener los Puntos de Función sin Ajustar (PFNA)
4. Obtener los Puntos de Función Ajustados (PFA)
5. Calcular el esfuerzo
6. Calcular duración del proyecto

## **2. Análisis del aplicativo**

Para el siguiente aplicativo, se considerará que todas las funciones identificadas serán de complejidad media.

### **Paso 1: Identificar componentes del sistema**

- ❖ Entradas Externas (EI): procesos en la que se introducen datos
- ❖ Salidas Externas (EO): procesos en donde se envían datos al exterior del sistema
- ❖ Consultas (EQ): procesos en donde se combinan un dato de entrada y uno de salida sin afectar a los almacenamientos La salida no contiene información derivada

- ❖ Archivos Externos (EIF): grupos de datos que se mantienen externamente
- ❖ Archivos Internos (ILF): grupos de datos relacionados entre sí internos al sistema

El sistema cuenta con:

Docentes:

- Registro de docentes(EI)
- Listado de todos los docentes (EO)
- Actualización de datos docente (EI)
- Búsqueda de docentes en específico (EQ)

Alumnos:

- Registro de alumnos (EI)
- Listado de todos los alumnos (EO)
- Actualización de datos alumno (EI)
- Búsqueda de alumnos en específico (EQ)

Asignaciones:

- Registro de asignaciones(EI)
- Listado de asignaciones(EO)
- Búsqueda de asignación específica (EQ)
- 1 tabla en BD (ILF)

Ficha tutoría:

- Registro de fichas de tutoría (EI)
- Listado de fichas de tutorías (EO)
- Búsqueda de ficha de tutoría en específico (EQ)
- 1 tabla en BD (ILF)

Sesiones tutoria:

- Registro de sesiones de tutorías(EI)
- Listado de sesiones (EO)
- 1 tabla en BD (ILF)

**Paso 2: Luego para cada componente se usan estas tablas para calcular su valor de función en base al de nro. de atributos que involucre y la cantidad de archivos que use (se multiplica la cantidad de componentes que existen de cada tipo por el valor correspondiente).**

<b>Clasificación de Entradas y Consultas</b>	<b>1 a 4 atributos SIMPLE</b>	<b>5 a 15 atributos MEDIA</b>	<b>Más de 15 atributos COMPLEJA</b>
Entradas externas(EI)	3	4	6
Archivos internos(ILF)	7	10	15
Consultas(EQ)	3	4	6

<b>Clasificación de Salidas</b>	<b>1 a 4 atributos SIMPLE</b>	<b>5 a 15 atributos MEDIA</b>	<b>Más de 15 atributos COMPLEJA</b>
Archivos externos(EIF)	5	7	10
Salidas externas(EO)	4	5	7

- Registro de docentes(EI 4 PF)
- Listado de todos los docentes (EO 5 PF)
- Actualización de datos docente (EI 4 PF)
- Búsqueda de docentes en específico (EQ 4 PF)
- Registro de alumnos (EI 4 PF)
- Listado de todos los alumnos (EO 5 PF)
- Actualización de datos alumno (EI 4 PF)
- Búsqueda de alumnos en específico (EQ 4 PF)



- Registro de asignaciones(EI 3 PF)
- Listado de asignaciones(EO 4 PF)
- Búsqueda de asignación específica (EQ 3 PF)
- Registro de fichas de tutoría (EI 3 PF)
- Listado de fichas de tutorías (EO 4 PF)
- Búsqueda de ficha de tutoría en específico (EQ 3 PF)
- Registro de sesiones de tutorías(EI 4 PF)
- Listado de sesiones (EO 5 PF)
- 3 tablas en BD (ILF 15 PF)

**Paso 3: Luego de obtenerse un valor para cada tipo de componentes del sistema sumando se calcula el PFNA (punto de función no ajustado)**

<b>Clasificación de Entradas y Consultas</b>	<b>1 a 4 atributos SIMPLE</b>	<b>5 a 15 atributos MEDIA</b>	<b>+15 atributos COMPLEJA</b>	<b>TOTAL</b>
Entradas externas(EI)	3 (hay 2)	4 (hay 5)	6 (no hay)	26
Archivos internos(ILF)	7 (hay 2)	10 (hay 1)	15 (no hay)	24
Consultas(EQ)	3 (hay 2)	4 (hay 2)	6 (no hay)	14

<b>Clasificación de Salidas</b>	<b>1 a 4 atributos SIMPLE</b>	<b>5 a 15 atributos MEDIA</b>	<b>+15 atributos COMPLEJA</b>	<b>TOTAL</b>
Archivos externos(EIF)	5 (no hay)	7 (no hay)	10 (no hay)	0
Salidas externas(EO)	4 (hay 2)	5 (hay 3)	7 (no hay)	23

$$\text{PFNA} = 26 + 24 + 14 + 0 + 23 = 87$$

## Paso 4: Obtener los Puntos de Función Ajustados (PFA)

Usando una tabla de factor de ajuste:

Factor de Ajuste	Puntaje
Comunicación de Datos	4
Procesamiento Distribuido	4
Objetivos de Rendimiento	1
Configuración del equipamiento	1
Tasa de transacciones	3
Entrada de Datos en Línea	5
Interfase con el usuario	2
Actualizaciones en Línea	3
Procesamiento Complejo	1
Reusabilidad del Código	1
Facilidad de Implementación	0
Facilidad de Operación	1
Instalaciones Múltiples	2
Facilidad de Cambios	4
<b>Factor de ajuste total</b>	<b>32</b>

Y la fórmula:

$$PFA = PFNA * [0.65 + (0.01 * \textit{factor de ajuste})]$$

Tenemos:

$$PFA = 87 * [0.65 + (0.01 * 32)]$$

$$PFA = 87 * [0.65 + 0.32]$$

$$PFA = 87 * [0.97]$$

$$PFA = 84.39$$

## Paso 5: Calcular el esfuerzo

El objetivo ahora es estimar la cantidad de esfuerzo necesario para desarrollar la aplicación. Este esfuerzo se mide en horas/hombre, meses/hombre o años/hombre.

Los puntos de función en cierto modo son una medida subjetiva. La cantidad de horas/hombre por punto de función es algo difícil e impreciso de valorar, de forma global. Esto es normal, lo contrario sería suponer que la productividad de todas las empresas de desarrollo de software es igual.

Para esto usaremos ahora una tabla con un estimado de líneas de código y horas promedio por punto de función:

Lenguaje	Horas PF promedio	Líneas de código por PF
Ensamblador	25	300
COBOL	15	100
Lenguajes de 4ta Generación (Javascript(React))	8	20

$$\frac{\text{Horas}}{\text{Hombre}} = PFA * \text{Horas PF promedio}$$

$$\frac{\text{Horas}}{\text{Hombre}} = 84.39 * 8$$

$$\frac{\text{Horas}}{\text{Hombre}} = 675.12$$

Por lo cual para este proyecto necesitamos aproximadamente 676 horas con un solo desarrollador.

## Paso 6: Calcular duración del proyecto

Estimando un equipo de desarrollo de 8 personas trabajando 2 horas diarias durante solo 10 días al mes tenemos:

$$Horas = \frac{675.12}{8} = 84.39 \leftarrow \text{Duración del proyecto en horas}$$

$$\frac{84.39}{2} = 42.195 \leftarrow \text{Días de trabajo}$$

$$\frac{42.195}{10} = 4.2195 \leftarrow 5 \text{ meses de trabajo}$$

### 3. Análisis no Funcional(ISO 25010)

#### a. Base Teórica

El modelo de calidad del producto definido por la ISO/IEC 25010 se encuentra compuesto por las ocho características de calidad que se muestran en la siguiente



figura:

#### b. Adecuación Funcional

Representa la capacidad del producto software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones especificadas. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Completitud funcional.** Grado en el cual el conjunto de funcionalidades cubre todas las tareas y los objetivos del usuario especificados.
- **Corrección funcional.** Capacidad del producto o sistema para proveer resultados correctos con el nivel de precisión requerido.
- **Pertinencia funcional.** Capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados.

#### c. Eficiencia de desempeño

Esta característica representa el desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Comportamiento temporal.** Los tiempos de respuesta y procesamiento y las ratios de throughput de un sistema cuando lleva a cabo sus funciones bajo condiciones determinadas en relación con un banco de pruebas establecido(benchmark).
- **Utilización de recursos.** Las cantidades y tipos de recursos utilizados cuando el software lleva a cabo su función bajo condiciones determinadas.
- **Capacidad.** Grado en que los límites máximos de un parámetro de un producto o sistema software cumplen con los requisitos.

#### **d. Compatibilidad**

Capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno hardware o software. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Coexistencia.** Capacidad del producto para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes sin detrimento.
- **Interoperabilidad.** Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.

#### **e. Usabilidad**

Capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Capacidad para reconocer su adecuación.** Capacidad del producto que permite al usuario entender si el software es adecuado para sus necesidades.
- **Capacidad de aprendizaje.** Capacidad del producto que permite al usuario aprender su aplicación.
- **Capacidad para ser usado.** Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.

- **Protección contra errores de usuario.** Capacidad del sistema para proteger a los usuarios de errores.
- **Estética de la interfaz de usuario.** Capacidad de la interfaz de usuario de agradar y satisfacer la interacción con el usuario.
- **Accesibilidad.** Capacidad del producto que permite que sea utilizado por usuarios con determinadas características y discapacidades.

## f. Fiabilidad

Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Madurez.** Capacidad del sistema para satisfacer las necesidades de fiabilidad en condiciones normales.
- **Disponibilidad.** Capacidad del sistema o componente de estar operativo y accesible para su uso cuando se requiere.
- **Tolerancia a fallos.** Capacidad del sistema o componente para operar según lo previsto en presencia de fallos hardware o software.
- **Capacidad de recuperación.** Capacidad del producto software para recuperar los datos directamente afectados y restablecer el estado deseado del sistema en caso de interrupción o fallo.

## g. Seguridad

Capacidad de protección de la información y los datos de manera que personas o sistemas no autorizados no puedan leerlos o modificarlos. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Confidencialidad.** Capacidad de protección contra el acceso de datos e información no autorizados, ya sea accidental o deliberadamente.
- **Integridad.** Capacidad del sistema o componente para prevenir accesos o modificaciones no autorizados a datos o programas de ordenador.

- **No repudio.** Capacidad de demostrar las acciones o eventos que han tenido lugar, de manera que dichas acciones o eventos no puedan ser repudiados posteriormente.
- **Responsabilidad.** Capacidad de rastrear de forma inequívoca las acciones de una entidad.
- **Autenticidad.** Capacidad de demostrar la identidad de un sujeto o un recurso.

## **h. Mantenibilidad**

Esta característica representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

- **Modularidad.** Capacidad de un sistema o programa de ordenador (compuesto de componentes discretos) que permite que un cambio en un componente tenga un impacto mínimo en los demás.
- **Reusabilidad.** Capacidad de un activo que permite que sea utilizado en más de un sistema software o en la construcción de otros activos.
- **Analizabilidad.** Facilidad con la que se puede evaluar el impacto de un determinado cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar.
- **Capacidad para ser modificado.** Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño.
- **Capacidad para ser probado.** Facilidad con la que se pueden establecer criterios de prueba para un sistema o componente y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.



## i. Portabilidad

Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno hardware, software, operacional o de utilización a otro. Esta característica se subdivide a su vez en las siguientes subcaracterísticas:

**Adaptabilidad.** Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.

**Capacidad para ser instalado.** Facilidad con la que el producto se puede instalar y/o desinstalar de forma exitosa en un determinado entorno.

**Capacidad para ser reemplazado.** Capacidad del producto para ser utilizado en lugar de otro producto software determinado con el mismo propósito y en el mismo entorno.

## Análisis del aplicativo

Adecuación Funcional	
Complejidad funcional	Si en un 88%
Corrección funcional	-
Pertinencia funcional	Si en un 90%

Eficiencia de desempeño	
Comportamiento temporal	<div>Usando <a href="https://developers.google.com/speed/pagespeed/insights/?hl=es">https://developers.google.com/speed/pagespeed/insights/?hl=es</a></div> <div>Prueba:</div> <div><div><div>50</div></div><div><a href="https://frontendtutoria.herokuapp.com/">https://frontendtutoria.herokuapp.com/</a></div><div><div><div>▲ 0-49</div><div>■ 50-89</div><div>● 90-100</div><div>i</div></div></div></div> <div><div>Datos de campo</div><div>— Chrome User Experience Report no tiene suficientes datos a tiempo real sobre la velocidad de esta página.</div></div> <div><div>Origin Summary</div><div>— Chrome User Experience Report no tiene suficientes datos a tiempo real sobre la velocidad de este origen.</div></div> <div><div>Datos de experimentos</div><div>— Las métricas se basan en datos de experimentos que ha recogido y analizado Lighthouse.</div></div> <div><div><div><div>▲ First Contentful Paint</div><div>7,7 s</div></div><div><div>▲ Time to Interactive</div><div>7,7 s</div></div></div><div><div><div>▲ Speed Index</div><div>9,1 s</div></div><div><div>● Total Blocking Time</div><div>10 ms</div></div></div><div><div><div>▲ Largest Contentful Paint</div><div>7,7 s</div></div><div><div>■ Cumulative Layout Shift</div><div>0,104</div></div></div></div> <div>Los valores son estimaciones y pueden variar. La puntuación del rendimiento se calcula directamente a partir de estas métricas. <a href="#">Ver calculadora.</a></div>
Utilización de recursos	<ul style="list-style-type: none"><li>● Heroku</li><li>● Azure</li><li>● Firebase</li></ul>
Capacidad	<ul style="list-style-type: none"><li>● Almacenamiento de 2 GB (limitación Azure + Sql)</li><li>● Uso de 550 a 1000 horas por mes (limitación heroku)</li></ul>

Compatibilidad	
Coexistencia	No tiene coexistencia con ningún otro sistema
Interoperabilidad	Se tiene una relación entre el backend, los cuales intercambian información dentro de la nube, por lo cual estos tienen la capacidad de interconectar el aplicativo

<b>Fiabilidad</b>	
<b>Madurez</b>	Posee la capacidad de responder todas las necesidades del usuario bajo condiciones normales.
<b>Disponibilidad</b>	Limitada por heroku al inicializar el aplicativo, pues este tiene un límite de horas
<b>Tolerancia a fallos</b>	Tiene una baja tolerancia a fallos:
<b>Capacidad de recuperación</b>	No, no se tienen copias de seguridad dentro de Azure

<b>Seguridad</b>	
<b>Confidencialidad</b>	Se establece un mecanismo sencillo de datos de las sesiones de tutoría, el cuál es manejado por el estudiante.
<b>Integridad</b>	No cuenta con integridad a la hora de proteger modificaciones dentro del aplicativo
<b>No repudio</b>	No se tiene un registro de las acciones o eventos realizados por el usuario
<b>Responsabilidad</b>	No puede rastrear las acciones de una entidad
<b>Autenticidad</b>	Se identifica a cada usuario del aplicativo

<b>Mantenibilidad</b>	
<b>Modularidad</b>	El sistema posee modularidad dentro de sus componentes, pero estos se encuentran desorganizados por carpetas según las necesidades de estos
<b>Reusabilidad</b>	El software viene a ser reusable, pues este es modificable por medio de la base de datos
<b>Analizabilidad</b>	Si es analizable, debido a la relación entre sus módulos

<b>Capacidad para ser modificado</b>	El código puede ser modificado, teniendo en cuenta la relación entre los módulos
<b>Capacidad para ser probado</b>	No tiene capacidad para pruebas, pues al ser un software coheso esto trae problemas

<b>Portabilidad</b>	
<b>Adaptabilidad</b>	Se limita al entorno de escritorio al no ser responsivo en su totalidad.
<b>Capacidad para ser instalado</b>	No requiere instalación
<b>Capacidad para ser reemplazado</b>	No tiene esta funcionalidad