



Patrones de diseño

Asignatura: Ing. de Software I

Docente: Roxana Lisette 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

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.controlles.js y este también a su vez llama a las consultas del archivo queries.js

```
Docentes.routes.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
23 router.delete('/docentes/:id',deleteDocenteById);
24 //funcion para recuperar
```

Docentes.routes.js

Docentes.controlles.js

```
src > controllers > Docentes.crontrollers.js > 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 addDocente=async (req,res)=>{
27   try{
28     const { nombre, apellido, categoria, email, celular, direccion }=req.body;
29     const pool=await getConnection();
30     const result=await pool.request().query(queries.addDocente(nombre, apellido, categoria, email, celular, direccion));
31     console.log("addDocente executed");
32     res.json(result.recordset);
33   }catch(error){
34     res.status(500);
35     res.send(error.message);
36   }
37 };
38 export const addDocentes=async (req,res)=>{
39   try{
40     const { nombres, apellidos, categorias, emails, celulares, direcciones }=req.body;
41     const pool=await getConnection();
42     const result=await pool.request().query(queries.addDocentes(nombres, apellidos, categorias, emails, celulares, direcciones));
43     console.log("addDocentes executed");
44     res.json(result.recordset);
45   }catch(error){
46     res.status(500);
47     res.send(error.message);
48   }
49 };
50 export const loginDocente=async (req,res)=>{
51   try{
52     const { usuario, contrasena }=req.body;
53     const pool=await getConnection();
54     const result=await pool.request().query(queries.loginDocente(usuario, contrasena));
55     console.log("loginDocente executed");
56     res.json(result.recordset);
57   }catch(error){
58     res.status(500);
59     res.send(error.message);
60   }
61 };
62 export const loginCoordinador=async (req,res)=>{
63   try{
64     const { usuario, contrasena }=req.body;
65     const pool=await getConnection();
66     const result=await pool.request().query(queries.loginCoordinador(usuario, contrasena));
67     console.log("loginCoordinador executed");
68     res.json(result.recordset);
69   }catch(error){
70     res.status(500);
71     res.send(error.message);
72   }
73 };
74 export const getTutorById=async (req,res)=>{
75   try{
76     const { id }=req.params;
77     const pool=await getConnection();
78     const result=await pool.request().query(queries.getTutorById(id));
79     console.log("getTutorById executed");
80     res.json(result.recordset);
81   }catch(error){
82     res.status(500);
83     res.send(error.message);
84   }
85 };
86 export const deleteDocenteById=async (req,res)=>{
87   try{
88     const { id }=req.params;
89     const pool=await getConnection();
90     const result=await pool.request().query(queries.deleteDocenteById(id));
91     console.log("deleteDocenteById executed");
92     res.json(result.recordset);
93   }catch(error){
94     res.status(500);
95     res.send(error.message);
96   }
97 };
98 export const updateDocenteById=async (req,res)=>{
99   try{
100     const { id, categoria, email, celular, direccion }=req.params;
101     const pool=await getConnection();
102     const result=await pool.request().query(queries.updateDocenteById(id, categoria, email, celular, direccion));
103     console.log("updateDocenteById executed");
104     res.json(result.recordset);
105   }catch(error){
106     res.status(500);
107     res.send(error.message);
108   }
109 };
110 export const getTutores=async (req,res)=>{
111   try{
112     const pool=await getConnection();
113     const result=await pool.request().query(queries.getTutores);
114     console.log("getTutores executed");
115     res.json(result.recordset);
116   }catch(error){
117     res.status(500);
118     res.send(error.message);
119   }
120 };
121 export const getCoordinador=async (req,res)=>{
122   try{
123     const pool=await getConnection();
124     const result=await pool.request().query(queries.getCoordinador);
125     console.log("getCoordinador executed");
126     res.json(result.recordset);
127   }catch(error){
128     res.status(500);
129     res.send(error.message);
130   }
131 };
132 export const getAllDocentes=async (req,res)=>{
133   try{
134     const pool=await getConnection();
135     const result=await pool.request().query(queries.getAllDocentes);
136     console.log("getAllDocentes executed");
137     res.json(result.recordset);
138   }catch(error){
139     res.status(500);
140     res.send(error.message);
141   }
142 };
143 export const getAllTutores=async (req,res)=>{
144   try{
145     const pool=await getConnection();
146     const result=await pool.request().query(queries.getAllTutores);
147     console.log("getAllTutores executed");
148     res.json(result.recordset);
149   }catch(error){
150     res.status(500);
151     res.send(error.message);
152   }
153 };
154 export const getAllCoordinadores=async (req,res)=>{
155   try{
156     const pool=await getConnection();
157     const result=await pool.request().query(queries.getAllCoordinadores);
158     console.log("getAllCoordinadores executed");
159     res.json(result.recordset);
160   }catch(error){
161     res.status(500);
162     res.send(error.message);
163   }
164 };
165 export const getAllEstudiantes=async (req,res)=>{
166   try{
167     const pool=await getConnection();
168     const result=await pool.request().query(queries.getAllEstudiantes);
169     console.log("getAllEstudiantes executed");
170     res.json(result.recordset);
171   }catch(error){
172     res.status(500);
173     res.send(error.message);
174   }
175 };
176 export const getAllMaterias=async (req,res)=>{
177   try{
178     const pool=await getConnection();
179     const result=await pool.request().query(queries.getAllMaterias);
180     console.log("getAllMaterias executed");
181     res.json(result.recordset);
182   }catch(error){
183     res.status(500);
184     res.send(error.message);
185   }
186 };
187 export const getAllAsignaturas=async (req,res)=>{
188   try{
189     const pool=await getConnection();
190     const result=await pool.request().query(queries.getAllAsignaturas);
191     console.log("getAllAsignaturas executed");
192     res.json(result.recordset);
193   }catch(error){
194     res.status(500);
195     res.send(error.message);
196   }
197 };
198 export const getAllCursos=async (req,res)=>{
199   try{
200     const pool=await getConnection();
201     const result=await pool.request().query(queries.getAllCursos);
202     console.log("getAllCursos executed");
203     res.json(result.recordset);
204   }catch(error){
205     res.status(500);
206     res.send(error.message);
207   }
208 };
209 export const getAllGrupos=async (req,res)=>{
210   try{
211     const pool=await getConnection();
212     const result=await pool.request().query(queries.getAllGrupos);
213     console.log("getAllGrupos executed");
214     res.json(result.recordset);
215   }catch(error){
216     res.status(500);
217     res.send(error.message);
218   }
219 };
220 export const getAllAlumnos=async (req,res)=>{
221   try{
222     const pool=await getConnection();
223     const result=await pool.request().query(queries.getAllAlumnos);
224     console.log("getAllAlumnos executed");
225     res.json(result.recordset);
226   }catch(error){
227     res.status(500);
228     res.send(error.message);
229   }
230 };
231 export const getAllDocentesPorCurso=async (req,res)=>{
232   try{
233     const { curso }=req.params;
234     const pool=await getConnection();
235     const result=await pool.request().query(queries.getAllDocentesPorCurso(curso));
236     console.log("getAllDocentesPorCurso executed");
237     res.json(result.recordset);
238   }catch(error){
239     res.status(500);
240     res.send(error.message);
241   }
242 };
243 export const getAllTutoresPorCurso=async (req,res)=>{
244   try{
245     const { curso }=req.params;
246     const pool=await getConnection();
247     const result=await pool.request().query(queries.getAllTutoresPorCurso(curso));
248     console.log("getAllTutoresPorCurso executed");
249     res.json(result.recordset);
250   }catch(error){
251     res.status(500);
252     res.send(error.message);
253   }
254 };
255 export const getAllCoordinadoresPorCurso=async (req,res)=>{
256   try{
257     const { curso }=req.params;
258     const pool=await getConnection();
259     const result=await pool.request().query(queries.getAllCoordinadoresPorCurso(curso));
260     console.log("getAllCoordinadoresPorCurso executed");
261     res.json(result.recordset);
262   }catch(error){
263     res.status(500);
264     res.send(error.message);
265   }
266 };
267 export const getAllEstudiantesPorCurso=async (req,res)=>{
268   try{
269     const { curso }=req.params;
270     const pool=await getConnection();
271     const result=await pool.request().query(queries.getAllEstudiantesPorCurso(curso));
272     console.log("getAllEstudiantesPorCurso executed");
273     res.json(result.recordset);
274   }catch(error){
275     res.status(500);
276     res.send(error.message);
277   }
278 };
279 export const getAllMateriasPorCurso=async (req,res)=>{
280   try{
281     const { curso }=req.params;
282     const pool=await getConnection();
283     const result=await pool.request().query(queries.getAllMateriasPorCurso(curso));
284     console.log("getAllMateriasPorCurso executed");
285     res.json(result.recordset);
286   }catch(error){
287     res.status(500);
288     res.send(error.message);
289   }
290 };
291 export const getAllAsignaturasPorCurso=async (req,res)=>{
292   try{
293     const { curso }=req.params;
294     const pool=await getConnection();
295     const result=await pool.request().query(queries.getAllAsignaturasPorCurso(curso));
296     console.log("getAllAsignaturasPorCurso executed");
297     res.json(result.recordset);
298   }catch(error){
299     res.status(500);
300     res.send(error.message);
301   }
302 };
303 export const getAllCursosPorCurso=async (req,res)=>{
304   try{
305     const { curso }=req.params;
306     const pool=await getConnection();
307     const result=await pool.request().query(queries.getAllCursosPorCurso(curso));
308     console.log("getAllCursosPorCurso executed");
309     res.json(result.recordset);
310   }catch(error){
311     res.status(500);
312     res.send(error.message);
313   }
314 };
315 export const getAllGruposPorCurso=async (req,res)=>{
316   try{
317     const { curso }=req.params;
318     const pool=await getConnection();
319     const result=await pool.request().query(queries.getAllGruposPorCurso(curso));
320     console.log("getAllGruposPorCurso executed");
321     res.json(result.recordset);
322   }catch(error){
323     res.status(500);
324     res.send(error.message);
325   }
326 };
327 export const getAllAlumnosPorCurso=async (req,res)=>{
328   try{
329     const { curso }=req.params;
330     const pool=await getConnection();
331     const result=await pool.request().query(queries.getAllAlumnosPorCurso(curso));
332     console.log("getAllAlumnosPorCurso executed");
333     res.json(result.recordset);
334   }catch(error){
335     res.status(500);
336     res.send(error.message);
337   }
338 };
339 export const getAllDocentesPorMateria=async (req,res)=>{
340   try{
341     const { materia }=req.params;
342     const pool=await getConnection();
343     const result=await pool.request().query(queries.getAllDocentesPorMateria(materia));
344     console.log("getAllDocentesPorMateria executed");
345     res.json(result.recordset);
346   }catch(error){
347     res.status(500);
348     res.send(error.message);
349   }
350 };
351 export const getAllTutoresPorMateria=async (req,res)=>{
352   try{
353     const { materia }=req.params;
354     const pool=await getConnection();
355     const result=await pool.request().query(queries.getAllTutoresPorMateria(materia));
356     console.log("getAllTutoresPorMateria executed");
357     res.json(result.recordset);
358   }catch(error){
359     res.status(500);
360     res.send(error.message);
361   }
362 };
363 export const getAllCoordinadoresPorMateria=async (req,res)=>{
364   try{
365     const { materia }=req.params;
366     const pool=await getConnection();
367     const result=await pool.request().query(queries.getAllCoordinadoresPorMateria(materia));
368     console.log("getAllCoordinadoresPorMateria executed");
369     res.json(result.recordset);
370   }catch(error){
371     res.status(500);
372     res.send(error.message);
373   }
374 };
375 export const getAllEstudiantesPorMateria=async (req,res)=>{
376   try{
377     const { materia }=req.params;
378     const pool=await getConnection();
379     const result=await pool.request().query(queries.getAllEstudiantesPorMateria(materia));
380     console.log("getAllEstudiantesPorMateria executed");
381     res.json(result.recordset);
382   }catch(error){
383     res.status(500);
384     res.send(error.message);
385   }
386 };
387 export const getAllMateriasPorMateria=async (req,res)=>{
388   try{
389     const { materia }=req.params;
390     const pool=await getConnection();
391     const result=await pool.request().query(queries.getAllMateriasPorMateria(materia));
392     console.log("getAllMateriasPorMateria executed");
393     res.json(result.recordset);
394   }catch(error){
395     res.status(500);
396     res.send(error.message);
397   }
398 };
399 export const getAllAsignaturasPorMateria=async (req,res)=>{
400   try{
401     const { materia }=req.params;
402     const pool=await getConnection();
403     const result=await pool.request().query(queries.getAllAsignaturasPorMateria(materia));
404     console.log("getAllAsignaturasPorMateria executed");
405     res.json(result.recordset);
406   }catch(error){
407     res.status(500);
408     res.send(error.message);
409   }
410 };
411 export const getAllCursosPorMateria=async (req,res)=>{
412   try{
413     const { materia }=req.params;
414     const pool=await getConnection();
415     const result=await pool.request().query(queries.getAllCursosPorMateria(materia));
416     console.log("getAllCursosPorMateria executed");
417     res.json(result.recordset);
418   }catch(error){
419     res.status(500);
420     res.send(error.message);
421   }
422 };
423 export const getAllGruposPorMateria=async (req,res)=>{
424   try{
425     const { materia }=req.params;
426     const pool=await getConnection();
427     const result=await pool.request().query(queries.getAllGruposPorMateria(materia));
428     console.log("getAllGruposPorMateria executed");
429     res.json(result.recordset);
430   }catch(error){
431     res.status(500);
432     res.send(error.message);
433   }
434 };
435 export const getAllAlumnosPorMateria=async (req,res)=>{
436   try{
437     const { materia }=req.params;
438     const pool=await getConnection();
439     const result=await pool.request().query(queries.getAllAlumnosPorMateria(materia));
440     console.log("getAllAlumnosPorMateria executed");
441     res.json(result.recordset);
442   }catch(error){
443     res.status(500);
444     res.send(error.message);
445   }
446 };
447 export const getAllDocentesPorAsignatura=async (req,res)=>{
448   try{
449     const { asignatura }=req.params;
450     const pool=await getConnection();
451     const result=await pool.request().query(queries.getAllDocentesPorAsignatura(asignatura));
452     console.log("getAllDocentesPorAsignatura executed");
453     res.json(result.recordset);
454   }catch(error){
455     res.status(500);
456     res.send(error.message);
457   }
458 };
459 export const getAllTutoresPorAsignatura=async (req,res)=>{
460   try{
461     const { asignatura }=req.params;
462     const pool=await getConnection();
463     const result=await pool.request().query(queries.getAllTutoresPorAsignatura(asignatura));
464     console.log("getAllTutoresPorAsignatura executed");
465     res.json(result.recordset);
466   }catch(error){
467     res.status(500);
468     res.send(error.message);
469   }
470 };
471 export const getAllCoordinadoresPorAsignatura=async (req,res)=>{
472   try{
473     const { asignatura }=req.params;
474     const pool=await getConnection();
475     const result=await pool.request().query(queries.getAllCoordinadoresPorAsignatura(asignatura));
476     console.log("getAllCoordinadoresPorAsignatura executed");
477     res.json(result.recordset);
478   }catch(error){
479     res.status(500);
480     res.send(error.message);
481   }
482 };
483 export const getAllEstudiantesPorAsignatura=async (req,res)=>{
484   try{
485     const { asignatura }=req.params;
486     const pool=await getConnection();
487     const result=await pool.request().query(queries.getAllEstudiantesPorAsignatura(asignatura));
488     console.log("getAllEstudiantesPorAsignatura executed");
489     res.json(result.recordset);
490   }catch(error){
491     res.status(500);
492     res.send(error.message);
493   }
494 };
495 export const getAllMateriasPorAsignatura=async (req,res)=>{
496   try{
497     const { asignatura }=req.params;
498     const pool=await getConnection();
499     const result=await pool.request().query(queries.getAllMateriasPorAsignatura(asignatura));
500     console.log("getAllMateriasPorAsignatura executed");
501     res.json(result.recordset);
502   }catch(error){
503     res.status(500);
504     res.send(error.message);
505   }
506 };
507 export const getAllAsignaturasPorAsignatura=async (req,res)=>{
508   try{
509     const { asignatura }=req.params;
510     const pool=await getConnection();
511     const result=await pool.request().query(queries.getAllAsignaturasPorAsignatura(asignatura));
512     console.log("getAllAsignaturasPorAsignatura executed");
513     res.json(result.recordset);
514   }catch(error){
515     res.status(500);
516     res.send(error.message);
517   }
518 };
519 export const getAllCursosPorAsignatura=async (req,res)=>{
520   try{
521     const { asignatura }=req.params;
522     const pool=await getConnection();
523     const result=await pool.request().query(queries.getAllCursosPorAsignatura(asignatura));
524     console.log("getAllCursosPorAsignatura executed");
525     res.json(result.recordset);
526   }catch(error){
527     res.status(500);
528     res.send(error.message);
529   }
530 };
531 export const getAllGruposPorAsignatura=async (req,res)=>{
532   try{
533     const { asignatura }=req.params;
534     const pool=await getConnection();
535     const result=await pool.request().query(queries.getAllGruposPorAsignatura(asignatura));
536     console.log("getAllGruposPorAsignatura executed");
537     res.json(result.recordset);
538   }catch(error){
539     res.status(500);
540     res.send(error.message);
541   }
542 };
543 export const getAllAlumnosPorAsignatura=async (req,res)=>{
544   try{
545     const { asignatura }=req.params;
546     const pool=await getConnection();
547     const result=await pool.request().query(queries.getAllAlumnosPorAsignatura(asignatura));
548     console.log("getAllAlumnosPorAsignatura executed");
549     res.json(result.recordset);
550   }catch(error){
551     res.status(500);
552     res.send(error.message);
553   }
554 };
555 export const getAllDocentesPorCursoYMateria=async (req,res)=>{
556   try{
557     const { curso, materia }=req.params;
558     const pool=await getConnection();
559     const result=await pool.request().query(queries.getAllDocentesPorCursoYMateria(curso, materia));
560     console.log("getAllDocentesPorCursoYMateria executed");
561     res.json(result.recordset);
562   }catch(error){
563     res.status(500);
564     res.send(error.message);
565   }
566 };
567 export const getAllTutoresPorCursoYMateria=async (req,res)=>{
568   try{
569     const { curso, materia }=req.params;
570     const pool=await getConnection();
571     const result=await pool.request().query(queries.getAllTutoresPorCursoYMateria(curso, materia));
572     console.log("getAllTutoresPorCursoYMateria executed");
573     res.json(result.recordset);
574   }catch(error){
575     res.status(500);
576     res.send(error.message);
577   }
578 };
579 export const getAllCoordinadoresPorCursoYMateria=async (req,res)=>{
580   try{
581     const { curso, materia }=req.params;
582     const pool=await getConnection();
583     const result=await pool.request().query(queries.getAllCoordinadoresPorCursoYMateria(curso, materia));
584     console.log("getAllCoordinadoresPorCursoYMateria executed");
585     res.json(result.recordset);
586   }catch(error){
587     res.status(500);
588     res.send(error.message);
589   }
590 };
591 export const getAllEstudiantesPorCursoYMateria=async (req,res)=>{
592   try{
593     const { curso, materia }=req.params;
594     const pool=await getConnection();
595     const result=await pool.request().query(queries.getAllEstudiantesPorCursoYMateria(curso, materia));
596     console.log("getAllEstudiantesPorCursoYMateria executed");
597     res.json(result.recordset);
598   }catch(error){
599     res.status(500);
600     res.send(error.message);
601   }
602 };
603 export const getAllMateriasPorCursoYMateria=async (req,res)=>{
604   try{
605     const { curso, materia }=req.params;
606     const pool=await getConnection();
607     const result=await pool.request().query(queries.getAllMateriasPorCursoYMateria(curso, materia));
608     console.log("getAllMateriasPorCursoYMateria executed");
609     res.json(result.recordset);
610   }catch(error){
611     res.status(500);
612     res.send(error.message);
613   }
614 };
615 export const getAllAsignaturasPorCursoYMateria=async (req,res)=>{
616   try{
617     const { curso, materia }=req.params;
618     const pool=await getConnection();
619     const result=await pool.request().query(queries.getAllAsignaturasPorCursoYMateria(curso, materia));
620     console.log("getAllAsignaturasPorCursoYMateria executed");
621     res.json(result.recordset);
622   }catch(error){
623     res.status(500);
624     res.send(error.message);
625   }
626 };
627 export const getAllCursosPorCursoYMateria=async (req,res)=>{
628   try{
629     const { curso, materia }=req.params;
630     const pool=await getConnection();
631     const result=await pool.request().query(queries.getAllCursosPorCursoYMateria(curso, materia));
632     console.log("getAllCursosPorCursoYMateria executed");
633     res.json(result.recordset);
634   }catch(error){
635     res.status(500);
636     res.send(error.message);
637   }
638 };
639 export const getAllGruposPorCursoYMateria=async (req,res)=>{
640   try{
641     const { curso, materia }=req.params;
642     const pool=await getConnection();
643     const result=await pool.request().query(queries.getAllGruposPorCursoYMateria(curso, materia));
644     console.log("getAllGruposPorCursoYMateria executed");
645     res.json(result.recordset);
646   }catch(error){
647     res.status(500);
648     res.send(error.message);
649   }
650 };
651 export const getAllAlumnosPorCursoYMateria=async (req,res)=>{
652   try{
653     const { curso, materia }=req.params;
654     const pool=await getConnection();
655     const result=await pool.request().query(queries.getAllAlumnosPorCursoYMateria(curso, materia));
656     console.log("getAllAlumnosPorCursoYMateria executed");
657     res.json(result.recordset);
658   }catch(error){
659     res.status(500);
660     res.send(error.message);
661   }
662 };
663 export const getAllDocentesPorCursoYAsignatura=async (req,res)=>{
664   try{
665     const { curso, asignatura }=req.params;
666     const pool=await getConnection();
667     const result=await pool.request().query(queries.getAllDocentesPorCursoYAsignatura(curso, asignatura));
668     console.log("getAllDocentesPorCursoYAsignatura executed");
669     res.json(result.recordset);
670   }catch(error){
671     res.status(500);
672     res.send(error.message);
673   }
674 };
675 export const getAllTutoresPorCursoYAsignatura=async (req,res)=>{
676   try{
677     const { curso, asignatura }=req.params;
678     const pool=await getConnection();
679     const result=await pool.request().query(queries.getAllTutoresPorCursoYAsignatura(curso, asignatura));
680     console.log("getAllTutoresPorCursoYAsignatura executed");
681     res.json(result.recordset);
682   }catch(error){
683     res.status(500);
684     res.send(error.message);
685   }
686 };
687 export const getAllCoordinadoresPorCursoYAsignatura=async (req,res)=>{
688   try{
689     const { curso, asignatura }=req.params;
690     const pool=await getConnection();
691     const result=await pool.request().query(queries.getAllCoordinadoresPorCursoYAsignatura(curso, asignatura));
692     console.log("getAllCoordinadoresPorCursoYAsignatura executed");
693     res.json(result.recordset);
694   }catch(error){
695     res.status(500);
696     res.send(error.message);
697   }
698 };
699 export const getAllEstudiantesPorCursoYAsignatura=async (req,res)=>{
700   try{
701     const { curso, asignatura }=req.params;
702     const pool=await getConnection();
703     const result=await pool.request().query(queries.getAllEstudiantesPorCursoYAsignatura(curso, asignatura));
704     console.log("getAllEstudiantesPorCursoYAsignatura executed");
705     res.json(result.recordset);
706   }catch(error){
707     res.status(500);
708     res.send(error.message);
709   }
710 };
711 export const getAllMateriasPorCursoYAsignatura=async (req,res)=>{
712   try{
713     const { curso, asignatura }=req.params;
714     const pool=await getConnection();
715     const result=await pool.request().query(queries.getAllMateriasPorCursoYAsignatura(curso, asignatura));
716     console.log("getAllMateriasPorCursoYAsignatura executed");
717     res.json(result.recordset);
718   }catch(error){
719     res.status(500);
720     res.send(error.message);
721   }
722 };
723 export const getAllAsignaturasPorCursoYAsignatura=async (req,res)=>{
724   try{
725     const { curso, asignatura }=req.params;
726     const pool=await getConnection();
727     const result=await pool.request().query(queries.getAllAsignaturasPorCursoYAsignatura(curso, asignatura));
728     console.log("getAllAsignaturasPorCursoYAsignatura executed");
729     res.json(result.recordset);
730   }catch(error){
731     res.status(500);
732     res.send(error.message);
733   }
734 };
735 export const getAllCursosPorCursoYAsignatura=async (req,res)=>{
736   try{
737     const { curso, asignatura }=req.params;
738     const pool=await getConnection();
739     const result=await pool.request().query(queries.getAllCursosPorCursoYAsignatura(curso, asignatura));
740     console.log("getAllCursosPorCursoYAsignatura executed");
741     res.json(result.recordset);
742   }catch(error){
743     res.status(500);
744     res.send(error.message);
745   }
746 };
747 export const getAllGruposPorCursoYAsignatura=async (req,res)=>{
748   try{
749     const { curso, asignatura }=req.params;
750     const pool=await getConnection();
751     const result=await pool.request().query(queries.getAllGruposPorCursoYAsignatura(curso, asignatura));
752     console.log("getAllGruposPorCursoYAsignatura executed");
753     res.json(result.recordset);
754   }catch(error){
755     res.status(500);
756     res.send(error.message);
757   }
758 };
759 export const getAllAlumnosPorCursoYAsignatura=async (req,res)=>{
760   try{
761     const { curso, asignatura }=req.params;
762     const pool=await getConnection();
763     const result=await pool.request().query(queries.getAllAlumnosPorCursoYAsignatura(curso, asignatura));
764     console.log("getAllAlumnosPorCursoYAsignatura executed");
765     res.json(result.recordset);
766   }catch(error){
767     res.status(500);
768     res.send(error.message);
769   }
770 };
771 export const getAllDocentesPorCursoYMateriaYAsignatura=async (req,res)=>{
772   try{
773     const { curso, materia, asignatura }=req.params;
774     const pool=await getConnection();
775     const result=await pool.request().query(queries.getAllDocentesPorCursoYMateriaYAsignatura(curso, materia, asignatura));
776     console.log("getAllDocentesPorCursoYMateriaYAsignatura executed");
777     res.json(result.recordset);
778   }catch(error){
779     res.status(500);
780     res.send(error.message);
781   }
782 };
783 export const getAllTutoresPorCursoYMateriaYAsignatura=async (req,res)=>{
784   try{
785     const { curso, materia, asignatura }=req.params;
786     const pool=await getConnection();
787     const result=await pool.request().query(queries.getAllTutoresPorCursoYMateriaYAsignatura(curso, materia, asignatura));
788     console.log("getAllTutoresPorCursoYMateriaYAsignatura executed");
789     res.json(result.recordset);
790   }catch(error){
791     res.status(500);
792     res.send(error.message);
793   }
794 };
795 export const getAllCoordinadoresPorCursoYMateriaYAsignatura=async (req,res)=>{
796   try{
797     const { curso, materia, asignatura }=req.params;
798     const pool=await getConnection();
799     const result=await pool.request().query(queries.getAllCoordinadoresPorCursoYMateriaYAsignatura(curso, materia, asignatura));
800     console.log("getAllCoordinadoresPorCursoYMateriaYAsignatura executed");
801     res.json(result.recordset);
802   }catch(error){
803     res.status(500);
804     res.send(error.message);
805   }
806 };
807 export const getAllEstudiantesPorCursoYMateriaYAsignatura=async (req,res)=>{
808   try{
809     const { curso, materia, asignatura }=req.params;
810     const pool=await getConnection();
811     const result=await pool.request().query(queries.getAllEstudiantesPorCursoYMateriaYAsignatura(curso, materia, asignatura));
812     console.log("getAllEstudiantesPorCursoYMateriaYAsignatura executed");
813     res.json(result.recordset);
814   }catch(error){
815     res.status(500);
816     res.send(error.message);
817   }
818 };
819 export const getAllMateriasPorCursoYMateriaYAsignatura=async (req,res)=>{
820   try{
821     const { curso, materia, asignatura }=req.params;
822     const pool=await getConnection();
823     const result=await pool.request().query(queries.getAllMateriasPorCursoYMateriaYAsignatura(curso, materia, asignatura));
824     console.log("getAllMateriasPorCursoYMateriaYAsignatura executed");
825     res.json(result.recordset);
826   }catch(error){
827     res.status(500);
828     res.send(error.message);
829   }
830 };
831 export const getAllAsignaturasPorCursoYMateriaYAsignatura=async (req,res)=>{
832   try{
833     const { curso, materia, asignatura }=req.params;
834     const pool=await getConnection();
835     const result=await pool.request().query(queries.getAllAsignaturasPorCursoYMateriaYAsignatura(curso, materia, asignatura));
836     console.log("getAllAsignaturasPorCursoYMateriaYAsignatura executed");
837     res.json(result.recordset);
838   }catch(error){
839     res.status(500);
840     res.send(error.message);
841   }
842 };
843 export const getAllCursosPorCursoYMateriaYAsignatura=async (req,res)=>{
844   try{
845     const { curso, materia, asignatura }=req.params;
846     const pool=await getConnection();
847     const result=await pool.request().query(queries.getAllCursosPorCursoYMateriaYAsignatura(curso, materia, asignatura));
848     console.log("getAllCursosPorCursoYMateriaYAsignatura executed");
849     res.json(result.recordset);
850   }catch(error){
851     res.status(500);
852     res.send(error.message);
853   }
854 };
855 export const getAllGruposPorCursoYMateriaYAsignatura=async (req,res)=>{
856   try{
857     const { curso, materia, asignatura }=req.params;
858     const pool=await getConnection();
859     const result=await pool.request().query(queries.getAllGruposPorCursoYMateriaYAsignatura(curso, materia, asignatura));
860     console.log("getAllGruposPorCursoYMateriaYAsignatura executed");
861     res.json(result.recordset);
862   }catch(error){
863     res.status(500);
864     res.send(error.message);
865   }
866 };
867 export const getAllAlumnosPorCursoYMateriaYAsignatura=async (req,res)=>{
868   try{
869     const { curso, materia, asignatura }=req.params;
870     const pool=await getConnection();
871     const result=await pool.request().query(queries.getAllAlumnosPorCursoYMateriaYAsignatura(curso, materia, asignatura));
872     console.log("getAllAlumnosPorCursoYMateriaYAsignatura executed");
873     res.json(result.recordset);
874   }catch(error){
875     res.status(500);
876     res.send(error.message);
877   }
878 };
879 export const getAllDocentesPorCursoYMateriaYAsignaturaYGrupo=async (req,res)=>{
880   try{
881     const { curso, materia, asignatura, grupo }=req.params;
882     const pool=await getConnection();
883     const result=await pool.request().query(queries.getAllDocentesPorCursoYMateriaYAsignaturaYGrupo(curso, materia, asignatura, grupo));
884     console.log("getAllDocentesPorCursoYMateriaYAsignaturaYGrupo executed");
885     res.json(result.recordset);
886   }catch(error){
887     res.status(500);
888     res.send(error.message);
889   }
890 };
891 export const getAllTutoresPorCursoYMateriaYAsignaturaYGrupo=async (req,res)=>{
892   try{
893     const { curso, materia, asignatura, grupo }=req.params;
894     const pool=await getConnection();
895     const result=await pool.request().query(queries.getAllTutoresPorCursoYMateriaYAsignaturaYGrupo(curso, materia, asignatura, grupo));
896     console.log("getAllTutoresPorCursoYMateriaYAsignaturaYGrupo executed");
897     res.json(result.recordset);
898   }catch(error){
899     res.status(500);
900     res.send(error.message);
901   }
902 };
903 export const getAllCoordinadoresPorCursoYMateriaYAsignaturaYGrupo=async (req,res)=>{
904   try{
905     const { curso, materia, asignatura, grupo }=req.params;
906     const pool=await getConnection();
907     const result=await pool.request().query(queries.getAllCoordinadoresPorCursoYMateriaYAsignaturaYGrupo(curso, materia, asignatura, grupo));
908     console.log("getAllCoordinadoresPorCursoYMateriaYAsignaturaYGrupo executed");
909     res.json(result.recordset);
910   }catch(error){
911     res.status(500);
912     res.send(error.message);
913   }
914 };
915 export const getAllEstudiantesPorCursoYMateriaYAsignaturaYGrupo=async (req,res)=>{
916   try{
917     const { curso, materia, asignatura, grupo }=req.params;
918     const pool=await getConnection();
919     const result=await pool.request().query(queries.getAllEstudiantesPorCursoYMateriaYAsignaturaYGrupo(curso, materia, asignatura, grupo));
920     console.log("getAllEstudiantesPorCursoYMateriaYAsignaturaYGrupo executed");
921     res.json(result.recordset);
922   }catch(error){
923     res.status(5
```

- 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 > .js 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

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

.env

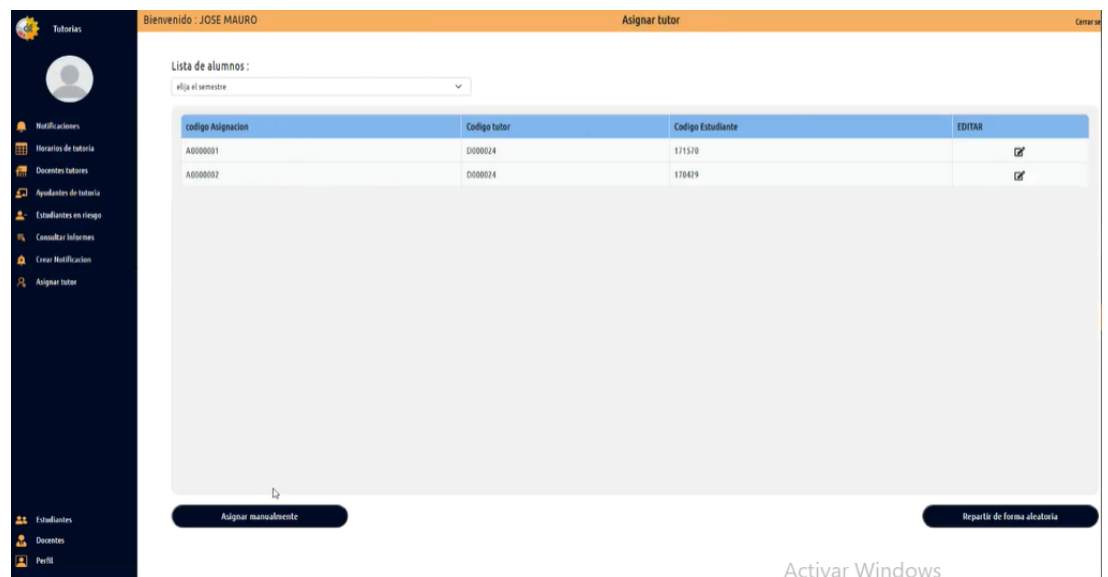
PATRONES DE DISEÑO EN FRONTEND

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

```
<Tutorbar/>
<div className="contenido">
  <div className="Principal2">
    <AdminBar nombrePage={"Estudiantes"} />
  </div>
  <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 peticionPostExcel=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.

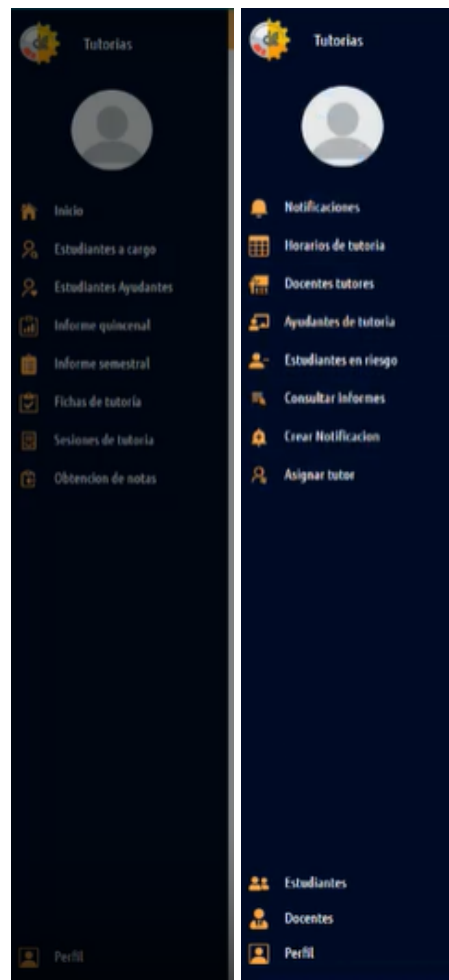
```
<ModalHeader>Asignar</ModalHeader>

<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>
  </Col>
</ModalBody>
```

- 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

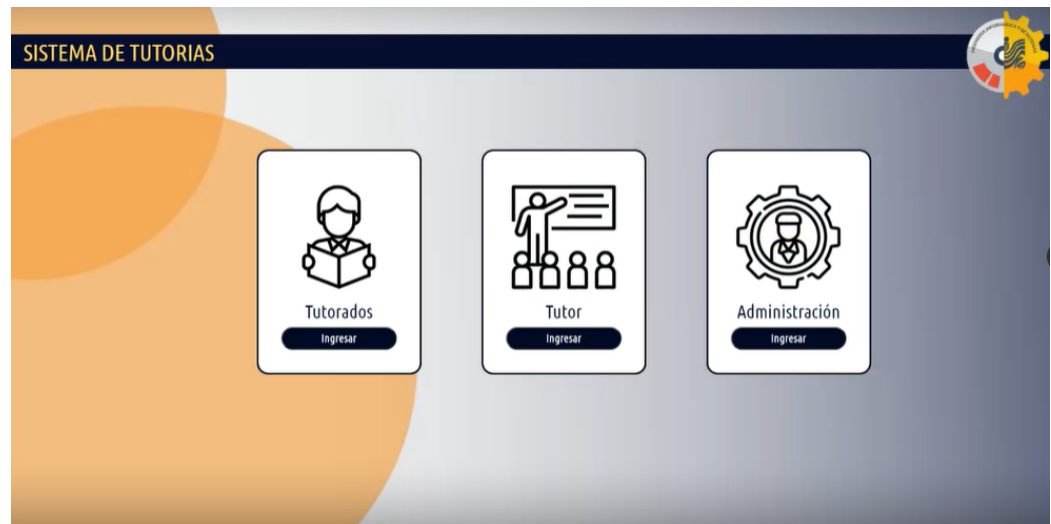
3. Patrones de comportamiento

- 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>