Proyecto Ingeniería de Software

1 Roles

Profesores:

- Sebastián Espinoza sespinoza@ubiobio.cl, evaluaciones de contenidos y evalúa Proyecto semestral.
- Alejandra Segura asegura@ubiobio.cl, clases y evaluaciones de contenidos.

Ayudantes:

- Coordinador: Diego Salazar, apoyo técnico en el STACK, clases online
- Product Owner:
  - Provee información del problema
  - Provee información para definir requisitos, alcances y límites, en representación del cliente.
  - Valida los requisitos definidos

Desarrolladores-Alumnos:

- Recopilar información a través de entrevistas, cuestionarios, observaciones e investigación
- Documentar técnicamente el desarrollo del sw
- Coordinar el trabajo del equipo
- Modelar procesos, datos y funcionalidad
- Codificar
- Probar el código implementado

2 Modelo de proceso del software

El enfoque de desarrollo iterativo e incremental por requisitos. Cada equipo será responsable de 1 proyecto, el cual considera cierta funcionalidad. Cada desarrollador será responsable de ciertos requerimientos de software.

- Avance backend
- Backend (implementación de cada requisito en el server, revisión de código e interrogación)
- Documentación del backend (avance en la documentación de la implementación y pruebas de cada requisito)
- Documentación del proyecto
- Software final (corresponde a la implementación front-end integrado al back end)

3 Organización Equipo

El equipo **se conforma de 3-4 integrantes**, tal que cada uno es responsable de 1 requisito.

A considerar:

- Si algún integrante del **equipo renuncia a la asignatura** se poblarán los datos en la base de datos de tal manera que los demás integrantes puedan probar sus funciones.
- Si algún integrante del **equipo no trabaja, trabajará solo. Cada parte será responsable solo de sus funciones,** se poblarán los datos en la base de datos de tal manera que los integrantes puedan probar sus funciones.

Cada equipo tiene asignado 1 product owner a quien representa al cliente.

4 Restricciones técnicas

- Stack MERN:
    - MongoDB- BD no relacional -- Mongoose
    - Frontend React /Next
    - Backend Express – postman insomnia

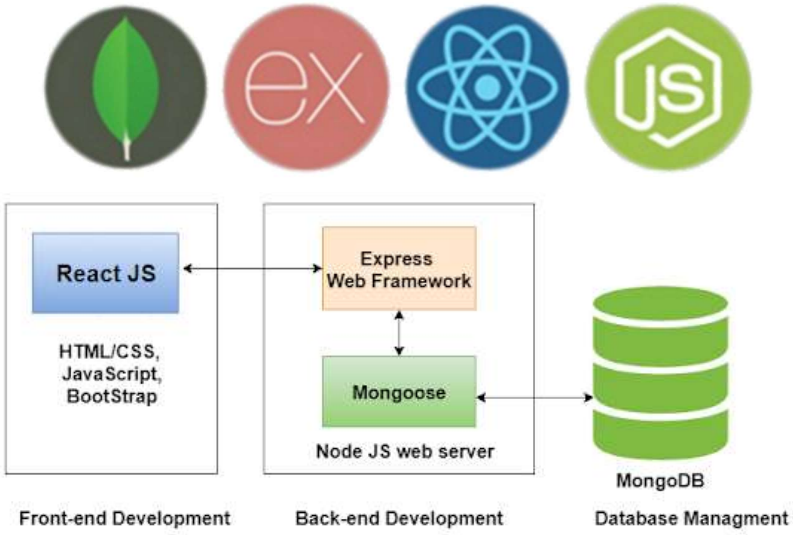

Ilustración 1: Ref https://blog.facialix.com/desarrollo-mern-stack/

- El proyecto debe estar alojado en el server de la FACE (solicitadas para cada equipo según distribución Linux/Ubuntu)—Server externo de respaldo
- Todo el código debe estar versionado en GIT de la FACE (son solicitadas para todo el curso) – Github UBB externo

5 Documentación técnica

La documentación técnica se compone de 2 entregables, cada uno de los cuales se encuentra estandarizado y debe ser completado siguiendo el estándar.

- Entrega Avance: Requerimientos, Análisis y Diseño. Según plantilla

- Manual de Desarrollo

6 Evaluaciones de Software

- Deben presentar un documento con documentación funcionalidades desarrolladas, credenciales de acceso a server apache /ssh, y usuarios con casos de prueba en la base de datos. En caso de back end, se detallan los endpoint implementados.
- La ejecución de la aplicación será en los **servidores de la FACE**. No se hacen evaluaciones en

máquinas locales o computadores personales.

- La revisión del proyecto de software será **solamente del código que pertenezca al repositorio grupal del proyecto, en la MASTER no en las ramas.**
- Integrantes que no tengan commit's en el repositorio, **serán evaluados con nota mínima**. Esto es determinante para conocer cuál fue su real aporte al desarrollo de la aplicación.
- Merge Request, solicitudes de integración y/o integraciones aceptadas/resueltas

6.1 Detalle de los ítems a evaluar

Evaluación Grupal:

1. Qué tan relacionado esta lo desarrollado con la temática principal de proyecto. Que lo diferencia de otros sistemas de información genéricos.
2. Qué tan relacionado esta lo desarrollado con los requerimientos propuestos por el equipo de desarrollo. ¿En qué medida se satisfacen los requerimientos en general?
3. Se aprecia una distribución equitativa de trabajo desarrollado por cada integrante del equipo. Capacidad de separar tareas.
4. Organización general, tanto a nivel de desarrollo (código ordenado, documentado), como a nivel de presentación (líneas de diseño, correcta visualización, etc.)
5. Organización general del equipo a la hora de presentar, así como conocimiento de los acuerdos y decisiones tomadas internamente.

Evaluación Individual:

1. Alumno desarrolla una cantidad suficiente de código, considerando tamaño del problema, requerimientos asignados, trabajo de sus compañeros, etc.
2. Alumno presenta de forma precisa y clara su software, considerando capacidad de síntesis, lenguaje utilizado y habilidades comunicativas.
3. Alumno demuestra conocimiento completo del código desarrollado y es capaz de responder preguntas de forma clara y precisa.
4. Alumno integra buenas prácticas de desarrollo de software, como validaciones de campos, usuarios y cualquier otra que, de cara al cliente, aporte calidad a su software.
5. Alumno presenta un software lo suficientemente bueno técnicamente según lo esperado en la asignatura. ¿Se satisface el requerimiento asignado? Se consideran, además, la evaluación de los puntos anteriores.

6.2 Puntaje de la rúbrica de evaluación

	No evaluable	Insuficiente	Medio	Suficiente	Excelente
Evaluación Grupal	**NCR**	**2,5**	**5**	**7,5**	**10**
Evaluación Individual	**NCR**	**5**	**10**	**15**	**20**

* Condiciones para NCR:

- No entregar código ejecutándose en servidor.
- No mostrar evidencias de funcionamiento.
- No realizar cambio solicitado
- Servidor o repositorio modificado fuera de fecha.
- Código insuficiente (No aporta como solución, demasiado simple, replica exacta de un compañero, etc.)

- **Puntaje máximo en evaluación grupal: 50**
- **Puntaje máximo en evaluación individual: 100**
- **Puntaje total: 150 = Nota 7 (Existencia 51%)**

6.3 Rubrica de evaluación de avance de software

Revisión Grupal

ID	Ítem	Puntaje	Comentario
1	Relación del tema con lo desarrollado		
2	Relación de los requerimientos con lo desarrollado		
3	Correcta distribución de esfuerzo		
4	Organización General del Software		
5	Organización General del equipo		

Observaciones individuales

ID	Ítem	Puntaje	Comentario	Cambio
Nombre:	Integrante 1			
1	Desarrolla su parte del software / Equivalente			SI
2	Presentación / Lenguaje / Comunicación			
3	Conocimiento de lo desarrollado / Preguntas			
4	Correcto Funcionamiento / Buenas Practicas / Calidad			
5	Evaluación general de las funcionalidades desarrolladas			
Nombre:	Integrante 2			
1	Desarrolla su parte del software / Equivalente			SI
2	Presentación / Lenguaje / Comunicación			
3	Conocimiento de lo desarrollado / Preguntas			
4	Correcto Funcionamiento / Buenas Practicas / Calidad			
5	Evaluación general de las funcionalidades desarrolladas			
Nombre:	Integrante 3			
1	Desarrolla su parte del software / Equivalente			SI
2	Presentación / Lenguaje / Comunicación			
3	Conocimiento de lo desarrollado / Preguntas			
4	Correcto Funcionamiento / Buenas Practicas / Calidad			
5	Evaluación general de las funcionalidades desarrolladas			
Nombre:	Integrante 4			
1	Desarrolla su parte del software / Equivalente			SI
2	Presentación / Lenguaje / Comunicación			
3	Conocimiento de lo desarrollado / Preguntas			
4	Correcto Funcionamiento / Buenas Practicas / Calidad			
5	Evaluación general de las funcionalidades desarrolladas			

Observaciones Extra

Puntaje Grupal	0

Puntaje Individual	Puntaje	Suma		
Integrante 1	0	0	Nota	1
Integrante 2	0	0	Nota	1
Integrante 3	0	0	Nota	1
Integrante 4	0	0	Nota	1

6.4 Buenas Prácticas – Mínimos que serán evaluadas en el software.

1. Documentación mínima de código
    a. bloques funcionales descritos
    b. identación
    c. nombres de variables y estructuras camel case
2. Dar un correcto formato a la información desplegada por las interfaces de usuario. Formato de fechas, horas, campos numéricos, información tabulada, descripciones de ID's para claves foráneas, etc
    a. Ej1: Las fechas deben mostrarse en formato DD-MM-AAAA y no en formato YYYY-MM-DD. Lo mismo opera para el ingreso de información en los formularios.
    b. Ej2: Sustituir ID's por las respectivas Descripciones, No se vale desplegar Estado: 1 , lo correcto es Estado: Registrado
3. Ingreso de datos utilizando drops - combo box - select de HTML según corresponda.
    a. Ej1: Si existen registros asociados a una clave foránea para el almacenamiento de la información para determinados formularios, se debe hacer referencia y mostrar la descripción asociada a la clave foránea y no simplemente mostrar los ID's, o un campo de texto/drop para que usuario ingrese ID's numéricos.
4. Para las estructuras/tablas de la Base de Datos, que tengan identificadores (Clave Primaria) numéricos, o autoincrementales, el ingreso de este campo no debe ser solicitado por la interfaz web al usuario.
5. Diseño de la Interfaz: uso de espacio, organización y nombres estandarizadas a lo largo de la aplicación
    a. UI unificada con el resto de integrantes del grupo.
    b. en la vista de datos de id no es relevante,
    c. priorizar que datos se ven y organizar la interfaz
    d. Incluir placeholder, label de ejemplos o microhelp, títulos descriptivos, migas de pan
6. Manejar los errores de ejecución/sistema de forma amigable para el usuario (No desplegar errores de Base de Datos, No desplegar errores o mostrar páginas en blanco).
7. Notificar por mensajes las acciones realizadas por el usuario. (Ej: Datos guardados correctamente, Información no válida, Falta ingresar X campo del formulario, Información Enviada Correctamente, etc)
8. Validación Campos de formularios: numérico, fechas, horas, strings, email, extensión y mime type de archivos adjuntos, campos requeridos no nulos, etc.
9. Capturar fechas, horas, estados de forma automática por el sistema.
    a. Ej1: cuando un formulario asociado a uno o más registros de la base de datos necesite guardar la fecha del ingreso del registro, este debe capturarse de forma automática. Cuidar o considerar la diferencia de hora del cliente y del server.
    b. Ej2: cuando en un formulario se ingresa información y el registro en la base de datos necesita guardarse con un estado inicial predeterminado, este no debe ser exigido al usuario.
10. Validaciones de campos, nombres, título o descripción; que deben ser texto, aunque pueden contener números. revisen expresiones regulares, al menos que antes del 1er espacio la

cadena no sea sólo texto. En estos campos se utilizan métodos de normalización del texto, para evitar que queden mal escritos.

   a. El formateo del rut VISUAL es independiente al formato en la BD, pueden incluir una máscara que puedan configurarle seteando automáticamente los puntos, QUE EL USUARIO no se preocupe si es con punto o sin punto, nosotros lo transformamos y presentamos y si es necesario lo volvemos a transformar para llevar a la BD
   b. Otro ejemplo es el Teléfono debe incluir una máscara para el formato, pero no permitir caracteres y validar mínimo y máx. caracteres, al menos.
   c. También el largo de los datos se restringe y validan fácilmente, es ideal indicar x caracteres y en la medida se escribe indicar caracteres restantes.
   d. Texto explicativo para las opciones, importante un ejemplo y además mencionar mínimos/máximos
   e. Título descriptivo de las interfaces q siempre recuerden al usuario que está haciendo

11. Falta de ortografías en interfaz: títulos, placeholder, text, botones, mensajes, etc.
12. Las listas deben estar ordenada, por ejemplo: las ultimas primero, o las primeras en vencer, ese criterio se indica claramente al usuario
13. Las listas NO deben ser sólo el vaciado de una tabla sin ninguna opción de filtro, opciones para ordenar por distintas columnas y buscar por campo o por texto
14. Listas desplegables siempre actualizadas desde la base de datos. Ejemplo: revise este código https://select2.org/tagging
15. Validación de formato de archivos de subida, tamaño y optimizar imágenes antes de guardar, consideren que deben ser livianas, hay librerías que optimizan las imágenes con tamaños y resolución.