|  |  |
| --- | --- |
| **UNIVERSIDAD NACIONAL, SEDE REGIONAL BRUNCA**  **EIF209 Programación IV**  **Prof: Ruben Mora Vargas**  **Prof: Juan Gamboa Abarca** |  |

**PROYECTO FINAL**

**“Desarrollo web moderno”**

|  |
| --- |
| **Objetivo** |

Fomentar la creatividad y autonomía del estudiantado mediante la definición y desarrollo de un proyecto informático de mediana complejidad. A partir de un arquetipo preestablecido y criterios claros (macro requisitos), los estudiantes propondrán la solución a implementar a un problema específico, abarcando la concepción, el diseño, el desarrollo y la puesta en marcha de una aplicación web con su infraestructura operativa en un entorno real, consolidando así los conocimientos adquiridos durante el curso.

|  |
| --- |
| **Aspectos generales** |

1. Lea y comprenda cuidadosamente lo que se le solicite.
2. El Proyecto puede ser realizado en grupos de 3 personas como máximo, según capacidad del proyecto, rol del estudiante y requerimientos, La cantidad de estudiantes es para aumentar el recurso, contar con soluciones propias y que se consiga un cumplimiento cercano al 100%.
3. El día de cada entrega, el proyecto será defendido de manera individual de acuerdo a su rol. La nota será mérito personal por lo que deben conocer todas las particularidades del sistema y tener pruebas de trabajo realizado en cuanto a documentación, Figma, archivos de configuración y repositorio Github.
4. Se deben respetar las etapas del proyecto mencionadas en el programa del curso:

Se trabaja con 4 etapas.

Base Inicial (10%): Se revisa la arquitectura y diseño del trabajo, diseño de estructura de datos y páginas web y la conexión con la API.

Aplicación Base (12%): Se revisan las funcionalidades del sistema final y se proponen cambios para la defensa final, se propone refactorización de código, y aplicación de seguridad.

Defensa Del Proyecto (14%): Se revisa la implementación del sistema, se muestra cómo responde al estrés de la aplicación con miles de datos, peticiones o usuarios, detalle de cumplimiento de requerimientos.

Nota: Si el proyecto no tiene un 80% del alcance solicitado, pierde la posibilidad de realizar la presentación final.

Presentación Final (14%): Se revisa la corrección de los últimos detalles que el profesor indica, realiza una presentación a los compañeros, profesor e invitados, presenta alcance, conclusiones y recomendaciones del proyecto.

1. Si se comprueba que existen dos o más proyectos similares o copiados, se procederá a colocar nota cero a todos los proyectos involucrados.
2. Se requiere acceso durante toda la revisión al repositorio de GitHub, donde está el proyecto, así mismo debe compartir una carpeta en drive con la copia final del código fuente como respaldo, y los archivos de documentación adicionales. Deben subir al Aula Virtual solo un archivo de texto con los dos enlaces solicitados (repositorio y documentación, ambos enlaces deben apuntar a repositorios compartidos con la cuenta RubenMoraAI en tiempo y forma.
3. Fechas importantes:
   1. Entrega del enunciado: lunes 24 de febrero del 2025
   2. Base Inicial: lunes 12 de mayo del 2025
   3. Aplicación Base: lunes 26 de mayo del 2025
   4. Defensa Del Proyecto: lunes 2 de junio del 2025
   5. Presentación Final: lunes 16 de junio del 2025

|  |
| --- |
| **Descripción del proyecto** |

El proyecto consiste en el desarrollo de una solución web que permita solucionar un problema real. Dicho problema debe venir del estudiante y corresponder a un problema real.

|  |
| --- |
| **Macro requerimientos general** |

El sistema debe tener como mínimo de acuerdo a los roles involucrados lo siguiente:

| **Requerimiento** | **Diseñador web** | **Frontend developer** | **Fullstack developer** | **Backend developer** |
| --- | --- | --- | --- | --- |
| **Identificación, Autenticación, Autorización** | Implementación de una librería de un tercero como Clerk, Firebase o similar. | Implementación de una librería de un tercero como Clerk, Firebase o similar. | Implementación de un sistema de autenticación desde cero desde la BD con JWT, envío de correos y flujo de recuperación de contraseña. | Implementación de un sistema de autenticación desde cero desde la BD con JWT y uso de autenticación con Google, correo y contraseña y passwordless, envío de correos y flujo de recuperación de contraseña. |
| **Diseño y Experiencia de Usuario (UI/UX)** | Landing page con animaciones 3D modernas, transiciones y tema claro/oscuro, dashboard principal con un sistema de diseño integral, Storybook relacionado, temas claro/oscuro y RWD, validación y optimización de métricas (PageSpeed Insights, Lighthouse). | Landing page con acabado limpio y moderno, transiciones y tema claro/oscuro (ej. TailwindCSS), dashboard principal con sistema de diseño integral, temas claro/oscuro y RWD, optimización de métricas. | Dashboard principal con sistema de diseño integral, temas claro/oscuro y RWD, optimización de métricas. | No aplica. |
| **Gestión de Datos y Operaciones CRUD** | Contribuir con la representación visual de datos (dashboards, gráficos, tablas) que facilite la interpretación de la información.Debe crear una visualización llamativa y agradable | Desarrollar interfaces para la manipulación de datos: formularios, listas, validaciones en el cliente y visualización dinámica. | Implementar operaciones CRUD completas que integren la lógica de negocio entre front-end y back-end, garantizando una experiencia coherente. Presentar los endpoints respectivos. | Diseñar el modelo de datos y desarrollar APIs para gestionar operaciones CRUD de forma segura y eficiente, asegurando la integridad de la información. Presentar los endpoints respectivos. |
| **Creación de Formularios Dinámicos y Flujos de Datos** | Diseñar interfaces de formularios intuitivas y coherentes, cuidando estética, usabilidad y accesibilidad. Colaborar en la definición del layout e interacción visual de formularios complejos o dinámicos para una experiencia intuitiva. | Implementar formularios interactivos con validaciones en el cliente y feedback inmediato (errores, mensajes de ayuda). Implementar formularios adaptativos que se modifiquen según la entrada del usuario, con manejo de estados, validación y feedback inmediato. | Coordinar validaciones en cliente y servidor para garantizar la integridad de los datos. Diseñar flujos completos que integren formularios, garantizando sincronización y coherencia en el intercambio de datos. | Implementar validaciones robustas en el servidor, asegurando la seguridad y consistencia de la información. Gestionar la lógica de procesamiento de datos de formularios, asegurando persistencia correcta y manejo adecuado de transacciones. |
| **Gestión y Modelado de Base de Datos (BD)** | No aplica | Asegurar que las interfaces se adapten a la estructura de datos para una correcta representación. | Participar en el diseño global del modelo de datos, asegurando coherencia entre la lógica de negocio y la BD. Presentar los endpoints respectivos. | Diseñar, implementar y optimizar el modelo de datos, garantizando integridad, escalabilidad, seguridad y eficiencia en las consultas. Presentar los endpoints respectivos. |
| **Integración con Servicios Externos / APIs Externas** | No aplica. | Integrar SDKs y consumir APIs de servicios externos, gestionando estados y errores en la interfaz. | Orquestar la comunicación entre el sistema interno y APIs externas, transformando datos según sea necesario. | Implementar y asegurar la comunicación con servicios de terceros, gestionando autenticaciones, transformaciones y errores. |
| **Desarrollo y Consumo de API REST/GraphQL** | No aplica. | Consumir APIs para obtener y mostrar datos de forma dinámica, manejando asincronía y actualizaciones en tiempo real. | Diseñar y consumir APIs que establezcan una comunicación estructurada entre cliente y servidor. | Desarrollar APIs seguras, documentadas y escalables, con control de versiones, manejo de errores y pruebas de carga. |
| **Testing y Calidad de Código** | Revisar la consistencia visual y realizar pruebas de usabilidad para asegurar el correcto comportamiento del diseño en distintos dispositivos. | Escribir pruebas unitarias y de integración para componentes de UI, validando en diversos navegadores y dispositivos. | Implementar pruebas end-to-end que validen la interacción entre front-end y back-end. | Desarrollar pruebas unitarias para la lógica del servidor, pruebas de seguridad y carga, y validar la estabilidad de los endpoints críticos. |
| **Optimización y Rendimiento** | Optimizar recursos visuales (imágenes, animaciones) y el layout para lograr tiempos de carga óptimos sin sacrificar la estética. | Optimizar el código, implementar lazy loading y reducir el tamaño de los bundles para mejorar la velocidad. | Coordinar estrategias de caching y optimización tanto en el cliente como en el servidor, asegurando una comunicación eficiente. | Optimizar consultas a la base de datos, aplicar caching a nivel de servidor y garantizar la escalabilidad y eficiencia en el uso de recursos. |
| **Seguridad y Protección de Datos** | Diseñar interfaces que minimicen errores de usabilidad y apoyen patrones seguros (ej. formularios con feedback claro y uso correcto de inputs). | Implementar validaciones en el cliente, sanitizar entradas y aplicar medidas contra ataques XSS y CSRF. | Garantizar la seguridad integral en la comunicación cliente-servidor, implementando encriptación y protocolos de seguridad en cada capa. | Desarrollar e implementar políticas de seguridad robustas, con encriptación en tránsito y en reposo, validaciones en el servidor y cumplimiento normativo (ej. GDPR). |
| **Implementación y Despliegue** | No aplica | Configurar procesos de build y despliegue para activos estáticos, garantizando carga óptima en producción. | Coordinar el despliegue integral (front-end y back-end), integrando pipelines CI/CD y configurando entornos de prueba y producción. | Gestionar la configuración del servidor, migraciones de base de datos, despliegue de servicios e integración en producción, garantizando estabilidad y escalabilidad. |
| **Accesibilidad y SEO** | Diseñar con pautas WCAG, utilizando estructuras semánticas y elementos visuales que favorezcan accesibilidad y SEO. | Implementar HTML semántico, ARIA y optimizar la estructura de la información para mejorar la visibilidad en buscadores. | Asegurar que la arquitectura permita prácticas SEO efectivas (ej. SSR, rutas amigables) sin descuidar la accesibilidad. | No aplica |
| **Notificaciones y Comunicación** | Diseñar componentes visuales para notificaciones (alertas, modales) coherentes con la identidad visual. | Implementar interfaces para notificaciones en tiempo real, usando WebSockets o librerías y gestionando su interacción. | Desarrollar mecanismos de notificación integrados en la aplicación (ej. WebSockets, email, SMS) y coordinar su comportamiento. | Desarrollar la lógica de envío de notificaciones, gestionar colas de mensajes e integrar servicios externos (SMS, email, push notifications). |
| **Documentación y Ayuda para el Usuario** | Crear guías visuales, prototipos y elementos gráficos (infografías, diagramas) que faciliten el uso de la aplicación. | Colaborar en la documentación de la interfaz, incluyendo comentarios, guías de usuario y ayudas contextuales (tooltips, modales). | Documentar integralmente la interacción entre componentes y generar manuales de uso y documentación técnica. | Elaborar documentación técnica detallada de APIs, modelos de datos, endpoints y protocolos de seguridad, facilitando mantenibilidad e integraciones futuras. |
| **Visualización de Datos, Reportes y Gráficos** | Diseñar dashboards y visualizaciones atractivas, definiendo paletas, tipografías y estilos que faciliten la interpretación. | Implementar componentes interactivos (charts, tablas, infografías) utilizando librerías de visualización, asegurando responsividad y experiencia óptima. | Integrar la visualización de datos en el flujo global, coordinando la actualización en tiempo real de reportes y gráficos. | Desarrollar endpoints que proporcionen datos en formatos adecuados para la visualización, optimizando consultas y garantizando entrega oportuna. |

|  |
| --- |
| **Requerimientos funcionales** |

Debe realizar una investigación, validación con el usuario y propuesta de los requerimientos funcionales de la aplicación de modo que sea acorde a los recursos del curso, el tiempo y el objetivo. Algunos detalles del requerimiento deben ser preguntados por el estudiante o bien definidos mediante una posible solución por parte de este, en este caso debe haber una validación de ideas por parte del profesor para evitar malentendidos a la entrega del proyecto. Esto para generar un ambiente normal de incertidumbre e información incompleta similar a situaciones cotidianas en las que se puede ver envuelto cualquier profesional.

**Dispone de 3 semanas** para la presentación, validación de requerimientos y presentación de la propuesta en firme de su proyecto al profesor, esta labor no tiene puntaje, pero es necesaria para realizar el proyecto. Dicho esto, debe haber una especificación de requerimientos por parte del estudiante con un formato de historias de usuario validada y autorizada por el profesor antes del 10 de marzo del 2025.

**Requisitos No Funcionales:**

1. La aplicación debe mostrar animaciones de carga mientras no tenga la información pertinente para el usuario.
2. La aplicación debe paginar la información para evitar cargas innecesarias al usuario y prever mecanismos para trabajar con conexiones lentas.
3. Se requiere animaciones cuando se realiza carga de archivos o procesos asíncronos o pesados por parte de la aplicación.
4. La aplicación debe estar disponible para acceder desde una computadora o desde un teléfono, por lo que se debe aplicar los principios de responsive web design
5. La aplicación debe desarrollarse bajo los patrones de diseño MVC y producer(API) – Consumer(Web application), así como otros que el estudiante considere y pueda justificar su utilización.
6. Control de cuentas de usuario por roles y logging de acciones de usuario en todo el proyecto.
7. Se requiere una cobertura del 100% del proyecto por medio de las pruebas unitarias utilizando Jest y Testing Library, si el recurso utilizado es fullstack developer o backend developer.
8. Estandarización de CSS mediante el diseño con buenas prácticas en tailwindcss. Utilización de Cspell para evitar errores ortográficos, eslint y pretty para ordenar y validar el código.
9. Desarrollar una API con Typescript, Next.js y Firebase para la autenticación, almacenamiento de información en Firestore y archivos en Cloud Storage en sus capas gratuitas.
10. Publicación del repositorio de manera automática mediante un repositorio en GitHub y vercel, en sus capas gratuitas.

Algunos de los puntos dependen los recursos utilizados y los requerimientos, consulte al profesor si aplica.

Adicionalmente se requiere la inclusión de IA generativa como funcionalidades extra para el usuario, en almenos 3 funciones.

No se aceptan Eccommerce o SaaS por que hay demasiados ejemplos en Internet y es fácil copiarlos y adaptarlos.

|  |
| --- |
| **Criterios de evaluación** |

Como directriz de la cátedra se establece como obligatoria la defensa del proyecto, por parte de todos los miembros del grupo, la nota será siempre tomada de manera individual.

La presente rúbrica permite evaluar los proyectos basados en los siguientes rubros:

|  |  |
| --- | --- |
| **Rubro** | **Valor** |
| 1. Base Inicial   Se revisa la arquitectura y diseño del trabajo, diseño de datos y páginas web y la conexión con la API. | **10%** |
| 1. Aplicación Base   Se revisan las funcionalidades del sistema final y se proponen cambios para la defensa final, refactorización de código, y seguridad. | **12%** |
| 1. Defensa Del Proyecto   Se revisa la implementación del sistema, Se realizan pruebas de estrés de la aplicación con muchos datos, Se valora el cumplimiento de los requerimientos y se revisan las pruebas de integración y procesos.  **Nota: Si el proyecto no tiene un 80% del alcance solicitado, pierde la posibilidad de realizar la presentación final.** | **14%** |
| 1. Presentación Final   Se corrigen los detalles que el profesor indica, se realiza una presentación a los compañeros, profesor e invitados, presenta alcance, conclusiones y recomendaciones del proyecto. | **14%** |
| **Total** | **50%** |

La escala de puntos indica el rango de puntos en los que puede ser puntuado cada uno de los criterios de la rúbrica según los avances o proyecto presentado por el estudiante.

1. **Base inicial(10 puntos).**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rubro** | **Alto (6 a 8)** | **Suficiente (2 a 5)** | **Bajo (0 a 1)** |
| Arquitectura y diseño del trabajo. | Realiza un esfuerzo alto por agregar y entender las relaciones de los componentes de la aplicación, lo demuestra en un diagrama de draw.io o similar | Hay falencias en la interpretación de los elementos y las relaciones de los componentes de la aplicación o el diagrama presenta pocos detalles a mejorar. | No realiza el trabajo, es vago en su realización o no cumple con lo solicitado |
| Diseño de Páginas web | Presenta un diseño completo de las principales pantallas, utiliza metáforas comunes del desarrollo web, demuestra interés y creatividad, el mockup es bastante completo. Conoce todas las posibles rutas | Presenta un diseño bastante bueno, faltan algunos detalles, rutas o páginas importantes, o los diseños son muy simples o enredados. | No demuestra interés en lograr el diseño, o lo realiza vagamente o tiene faltantes más de dos páginas importantes. |
| Diseño de datos | Presenta un diseño completo de las principales entidades, documentos, atributos y sus relaciones, demuestra interés, análisis y creatividad, el diseño es bastante completo. Conoce todas las posibles rutas | Presenta un diseño bastante bueno, faltan algunos detalles, atributos o relaciones, o los diseños son muy simples o enredados. | No demuestra interés en lograr el diseño, o lo realiza vagamente, o tiene faltantes más de dos entidades importantes. |
| Conexión con la API. | Se desarrolla la API del sistema, trabaja con los módulos de base de datos y se consume desde la aplicación web | Solo existe la API, está bien formada y trabaja con la base de datos, tiene pocos faltantes o presenta algún bug menor. | No existe la API, está incompleta, o no está bien formada |

1. **Aplicación Base (12 puntos).**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rubro** | **Alto (6 a 8)** | **Suficiente (2 a 5)** | **Bajo (0 a 1)** |
| Funcionamiento integral de la aplicación web | Se desarrolló el aplicativo completo. Puede que le falten detalles menores a mejorar o detalles de diseño u optimización. | Se desarrollaron parte de los elementos/ requerimientos del aplicativo solicitado, pero queda pendiente gran parte del mismo. Lo que se hizo funciona al menos al 50%, o presenta varios errores considerables. | Existe más de un 50% de faltantes en la aplicación, o no funcionan correctamente, o están mal estructuradas, o no tienen control de errores. |
| API Funcional y datos bien formados | Presenta una API completa, robusta, bien estructurada y funcional, tiene seguridad, está cubierta por pruebas unitarias, y requiere pocas refactorizaciones | Presentan una API casi completa con faltantes en optimización o falta manejo completo de errores. | No existe la API, está en estado básico o tiene faltantes para el manejo de entidades o errores. |
| Manejo de validaciones y pruebas unitarias. | Todo el código en TypeScript de la aplicación generado está cubierto al 100% por pruebas unitarias. | La aplicación muestra al menos un 60% de las pruebas unitarias importantes. | La aplicación cuenta con pocas o nulas pruebas unitarias. |

1. **Defensa Del Proyecto (14 puntos)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rubro** | **Alto (6 a 8)** | **Suficiente (2 a 5)** | **Bajo (0 a 1)** |
| Funcionamiento integral de la aplicación web | Se desarrolló el aplicativo completo. Se corrigen los detalles de la evaluación anterior | Se desarrollaron parte de los elementos/ requerimientos del aplicativo solicitado, pero queda pendiente gran parte del mismo. Lo que se hizo funciona al menos al 70%, o presenta varios errores considerables. | Existe más de un 30% de faltantes en la aplicación, o no funcionan correctamente, o están mal estructuradas, o no tienen control de errores. |
| API Funcional y datos bien formados | Presenta una API completa, robusta, bien estructurada y funcional, tiene seguridad y refactorizada | Presentan una API casi completa(80%) con faltantes en optimización o falta manejo completo de errores. | No existe la API, está en estado básico o tiene faltantes para el manejo de entidades o errores. |
| Manejo de validaciones y pruebas unitarias. | Todo el código en TypeScript de la aplicación generado está cubierto al 100% por pruebas unitarias. | La aplicación muestra al menos un 60% de las pruebas unitarias importantes. | La aplicación cuenta con pocas o nulas pruebas unitarias. |
| Pruebas de estrés | El sistema cuenta con mecanismos para trabajar con volúmenes de datos altos, o mejorar la carga al usuario, y los estudiantes muestran este funcionamiento según lo sugerido. | El sistema presenta algunas pruebas o estas fallan, se realiza un muy buen intento por parte de los estudiantes. | No hay acciones para mejorar el rendimiento de la aplicación, o su carga, o no se aplican correctamente, o el esfuerzo es insuficiente. |

1. **Presentación Final (14 puntos).**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rubro** | **Alto (6 a 8)** | **Suficiente (2 a 5)** | **Bajo (0 a 1)** |
| Presentación del proyecto | Presentan un proyecto basado en objetivos y soluciones y conclusiones, involucran a los compañeros y atienden todas las dudas señaladas. | Presenta el proyecto de una manera básica, atienden las dudas señaladas o la presentación es aburrida o no involucra al público. | No presentan el proyecto, no está bien estructurada la presentación, o no atienden dudas o no las pueden responder. |
| Recomendaciones y conclusiones | Presenta una documentación completa del proyecto e incluye recomendaciones y conclusiones importantes. | Presenta la documentación completa y las recomendaciones y conclusiones no son significativas. | No presenta documentación, o no presenta conclusiones o recomendaciones. |
| Corrigieron todos los detalles señalados en la sesión anterior. | Se corrigen todas las recomendaciones dadas, presentan un código refactorizado y limpio | Solo existe la API, está bien formada y trabaja con Firebase, tiene pocos faltantes o presenta algún bug menor. | No existe la API, está incompleta, o no está bien formada |