# Modelo Dreyfus - Desarrollador Angular

PDF **Modelo dreyfus - Angular test.pdf** 554.66 kB

## Planificación de Proyectos

Explica cómo planificarías un proyecto en Angular, incluyendo pasos como definir el alcance del proyecto, establecer hitos y asignar recursos.

> *Para planificar un proyecto en Angular buscando tener un plan integral, minimizando riesgos y maximizando la eficiencia del equipo yo seguiria un enfoque claro y estructurado que abarque desde la definición del alcance del proyecto hasta la asignación de recursos, estableciendo las siguientes fases:*
>
> - ***Definir el alcance del proyecto*** *identificando y recopilando los requisitos de este, funcionalidades, objetivos comerciales y expectativas de los interesados, para posteriormente elaborar un documento que detalle el alcance del proyecto, sus objetivos, los entregables previstos, restricciones y criterios de éxito. Y finalmente presenta este documento de alcance a los interesados para su revisión y aprobación, asegurándome de recibir su aprobación y entendimiento.*
> - ***Descomposición del proyecto*** *de tal manera que este se divida en fases más manejables que permitan definir hitos significativos para cada fase.*
> - ***Creación de cronograma*** *Utilizando herramientas de gestión de proyectos para crear un cronograma que muestre la secuencia de hitos y actividades, asignando plazos realistas.*
> - ***Revisión con el equipo*** *con el fin de obtener su retroalimentación, ajustando si es necesario para garantizar la viabilidad.*
> - ***Identificación de los recursos necesarios*** *como desarrolladores, diseñadores, y hardware o infraestructura.*
> - ***Asignación de tareas*** *específicas a los miembros del equipo, considerando sus habilidades y disponibilidad.*
> - ***Monitoreo de Recursos*** *estableciendo un método para efectivo para medir la utilización de recursos a lo largo del proyecto, asegurando su disponibilidad oportuna.*

## Gestión de Dependencias

¿Cómo manejas las dependencias en un proyecto de Angular?

Describe el proceso para gestionar bibliotecas y dependencias de terceros de manera eficiente.

*Para manejar de manera eficiente bibliotecas y dependencias de terceros en un proyecto angular me apoyaria principalmente en el uso de la Interfaz De Linea De Comandos De Angular (Angular CLI) en conjunto de un gestor de paquetes como NPM o YARN para asi aprovechar los comando que este me proporciona para agregar, actualizar y/o eliminar dependencias , mantener un control preciso de las versiones de las dependencias y administrar las actualizaciones de estas de manera segura gracias al gestor de paquetes, todo esto combinado con una configuracion adecuada de las dependencias en el archivo* ***angular.json*** *del proyecto para garantizar que sean gestionadas de manera eficiente, asi como un correcto uso de mecanismos que el framework provee como lo es la inyeccion de dependencias. Todo esto combinado con el uso de buenas practicas como lo es la revision periodica de nuevas version y/o parches de seguridad para determinar la actualizacion o no de una dependencia.*

## Colaboración en Equipo

Habla sobre estrategias para una colaboración efectiva dentro de un equipo de desarrollo Angular. ¿Cómo garantizarías una comunicación fluida y coordinación entre los miembros del equipo?

*Creo que para fomentar una colaboración efectiva dentro de un equipo de desarrollo de software y garantizar una comunicación fluida y una coordinación óptima entre los miembros del equipo es crucial implementar estrategias tales como:*

- *Definir y utiliza herramientas de comunicación, preferiablemente con enfoque empresarial como Slack o Microsoft Teams para facilitar la interacción en tiempo real entre los miembros del equipo. Asi como tambien, crear canales específicos para discutir solo los temas relacionados a estos, como por ejemplo: problemas técnicos, actualizaciones, entre otros.*

- *Programa reuniones periódicas para revisar el progreso del proyecto, discutir obstáculos y compartir actualizaciones tratando de asegurar que estas reuniones sean inclusivas y que todos los miembros del equipo tengan la oportunidad de expresar sus ideas y preocupaciones.*

- *Apoyarse en el uso de herramientas como Jira, Trello o Asana para asignar tareas, hacer un seguimiento del progreso y gestionar el flujo de trabajo de manera efectiva, y aplicar buenas practicas como actualizar regularmente el estado de las tareas para mantener a todos informados y sincronizados.*

- *Fomentaria la colaboración y transferencia de conocimientos dentro del equipo implementando prácticas como **Pair Programming**, donde dos desarrolladores trabajan en conjuntos sobre el mismo código, compartiendo conocimientos y habilidades.*

- *Implementaria revisiones de código regularmente para garantizar la consistencia, calidad y buenas prácticas de codificación apoyandome de la utilizacion de herramientas como GitHub o gitLab para facilitar el proceso de revisión y asegurar que todos los miembros estén al tanto de los cambios.*

- *Creo que la retroalimentación ayuda a mejorar el rendimiento individual y colectivo, y a fortalecer la colaboración en el equipo por lo que fomentaria una cultura de retroalimentación constructiva donde los miembros del equipo se sientan cómodos dando y recibiendo comentarios.*

# Priorización de Tareas

> ℹ Supongamos que te encuentras con tareas conflictivas en un proyecto Angular. ¿Cómo priorizarías las tareas para garantizar una entrega oportuna y cumplir con los plazos del proyecto?

*Para garantizar una entrega oportuna y cumplir con los plazos del proyecto, yo me apegaria a un enfoque estructurado, teniendo en cuenta:*

- ***El impacto en el producto**: Evalúar el impacto que cada tarea tendrá en el producto final, priorizando aquellas tareas que tienen un impacto directo en la funcionalidad principal o en los requisitos fundamentales.*
- ***La urgencia**: Considerar el grado de urgencia de cada tarea. Abordando con prioridad aquellas que afecten la estabilidad, seguridad o usabilidad.*
- ***Identificar Dependencias**: Analizar si existen dependencias entre las tareas. Abordando con prioridad aquellas que actúen como requisitos previos para otras.*
- ***Consenso del Equipo**: Reúnir al equipo para discutir y validar la importancia de cada tarea teniendo en cuenta la retroalimentación del equipo, ya que puede ayudar a obtener una perspectiva más completa y a identificar posibles problemas que podrían haber sido pasados por alto.*
- ***Enfoque en el Valor para el Usuario**: Priorizar aquellas tareas que agreguen un valor significativo para el usuario final, como puede ser, incluir nuevas características, corrección de errores críticos o mejoras de rendimiento perceptibles.*
- ***Estimaciones realistas**: Asegúrarse de contar con estimaciones realistas de la complejidad y el esfuerzo requerido para cada tarea ya que la priorización debe considerar la viabilidad de completar la tarea dentro del plazo establecido.*

- ***Reevaluación Continua**: Ya que la priorización es un proceso dinámico, revisar y reevalúar constantemente la prioridad de las tareas a medida que avanza el proyecto permite obtener nueva información.*

# Control de Versiones

> Explica la importancia del control de versiones en proyectos de desarrollo Angular.
> Describe cómo usarías Git o cualquier otro sistema de control de versiones para gestionar cambios de código de manera efectiva.
> Explica como abordas la gestión y manejo de resolución de conflictos en ramas.

- *El control de versiones es crucial e importante en proyectos de desarrollo debido a que:*
  - *Facilita el trabajo colaborativo al permitir a varios desarrolladores trabajar en diferentes partes del código simultáneamente*
  - *Permite el seguimiento de los cambios en el código, lo que facilita la identificación de errores y la reversión a versiones anteriores si es necesario.*
  - *Ayuda a mantener un historial de cambios, lo que es fundamental para la trazabilidad y la auditoría del código.*
  - *Permite la implementación de buenas prácticas de desarrollo, como ramificación y fusión, para una gestión efectiva del código sin afectar el trabajo de otros desarrolladores.*

- *Para gestionar cambios de código de manera efectiva con Git en un proyecto de desarrollo, aplicaria una serie de practicas como:*
  - *Creacion de ramas separadas para cada función o tarea a desarrollar, permitiendo un trabajo aislado y paralelo.*
  - *Realizar commits de forma regular y significativa con mensajes descriptivos para documentar los cambios realizados.*
  - *Realizar Fusiones limpias de las ramas de desarrollo con la rama principal para integrar los cambios de manera ordenada.*
  - *Utilizar pull requests para revisar y discutir los cambios antes de fusionar una rama con la principal, asegurando la calidad del código*
  - *En caso de conflictos durante la fusión, abordar la resolución combinando los cambios manualmente o utilizando herramientas de resolución de conflictos integradas en Git.*

- *Para abordar la gestión y manejo de resolución de conflictos en ramas de manera efectiva en proyectos de desarrollo fometaria y aplicaria buenas practicas como :*
  - *Mantener una comunicación abierta con el equipo para entender el origen de los conflictos y buscar soluciones consensuadas.*

- *Analizar cuidadosamente los conflictos presentados por Git, identificando las diferencias y para decidir cómo resolverlos.*
- *En caso de conflictos manuales, revisaría el código en conflicto, decidir qué cambios mantener y llevaría a cabo la resolución manualmente.*
- *Utilizar herramientas de comparación y resolución de conflictos, como herramientas visuales o extensiones de Git, para facilitar el proceso de resolución en casos más complejos.*
- *Después de la resolución de conflictos, realizar pruebas exhaustivas para garantizar que el código integrado funcione correctamente sin problemas derivados de la resolución.*

# Manejo de Errores

ℹ Cómo abordarías el manejo de errores en una aplicación Angular? Habla sobre técnicas para identificar, registrar y resolver errores para mantener la estabilidad del proyecto.

*En pro de manterner la estabilidad del proyecto, yo usaria las siguientes técnicas para identificar, registrar y resolver errores de manera efectiva:*

- *Durante la depuracion de errores utiliza la consola del navegador para identificar errores de JavaScript y/o de Angular.*
- *Durante el desarrollo implementa el manejo de errores en Angular utilizando constructos como try-catch y los operadores catchError en las peticiones HTTP, para capturar y gestionar errores en partes especificas delcódigo.*
- *Implementar el uso de herramientas como Sentry, Rollbar o New Relic para monitorear en tiempo real los errores en la aplicación Angular ya que estas herramientas proporcionan información detallada sobre los errores que ocurren en producción.*
- *Implementar un sistema de registro de errores para registrar información detallada sobre los errores que ocurren en la aplicación, incluyendo contexto, ubicación y datos relevantes que no necesariamente se obtendrían de manera automática a través de herramientas de monitoreo externas.*
- *Utilizar servicios de centralización de logs como Loggly, Logstash o ELK stack para consolidar y analizar los registros de errores, lo que facilita la identificación de patrones y la resolución de problemas recurrentes.*
- *Implementar pruebas unitarias y de integración para detectar y prevenir errores durante el desarrollo.*
- *Utilizar versionamiento semántico y herramientas como Semantic Versioning para identificar de manera clara los cambios que corrigen errores en la aplicación.*
- *Implementar mecanismos para que los usuarios puedan reportar errores de forma sencilla, lo que permitirá abordar los problemas de manera proactiva.*
- *Realizar un analisis retrospectivo para entender a fondo los errores críticos, identificar sus causas subyacentes y tomar medidas para prevenir su recurrencia en el futuro.*

- *Basandose en el analisis retrospectivo realizar refactorizaciones proactivas y optimizar áreas propensas a errores en el código.*

## Pruebas y Aseguramiento de la Calidad

Describe tu enfoque para probar aplicaciones Angular.
¿Cómo garantizas la calidad del código, realizas pruebas unitarias y llevas a cabo pruebas de extremo a extremo para un proyecto?

*Creo que es relativo a las necesidad y requerimientos del cliente, asi como el ambito y alcance del proyecto mismo, sin embargo creo que como minimo se debe contar con pruebas unitarias donde facilmente se podria usar el framework de pruebas unitarias de Angular, para escribir pruebas que validen el comportamiento de componentes, servicios y directivas de manera aisladas, acompañado de la implementacion de mocks para simular dependencias y mantener la independencia de las pruebas unitarias. Todo esto de la mano de buenas practicas y altos estándares de calidad de código validandolo con linters como ESLint y herramientas de inspección de código estático.*

*Si el alcance lo permite emplearía pruebas de integración utilizando herramientas como Karma para evaluar la interacción entre múltiples componentes en un entorno simulado del navegador y verificaría la integración correcta de componentes, servicios y módulos dentro de la aplicación. Tambien implementaria pruebas de extremo a extremo, haciendo uso por ejemplo de herramientas para este fin, como por ejemplo Protractor que una herramienta de automatización de pruebas E2E diseñada específicamente para aplicaciones Angular, con la que escenificaría interacciones del usuario real para validar el flujo de la aplicación desde el inicio hasta el final.*

## Estrategias de Contenerización

Describe diferentes estrategias de contenerización para aplicaciones Angular. Habla sobre consideraciones como la gestión de variables en diferentes entornos.

*Algunas estrategias de contenerización para aplicaciones Angular, junto con consideraciones para la gestión de variables en diferentes entornos, incluyen:*

### *1. Contenedores Docker Independientes para Diferentes Entornos*

- ***Descripción**: Crear imágenes de contenedores Docker independientes para cada entorno, como desarrollo, pruebas y producción.*
- ***Consideraciones**:*

- *Utilizar variables de entorno específicas dentro de cada contenedor para configurar la aplicación según las necesidades de cada entorno.*
- *Emplear herramientas de orquestación como Docker Compose para gestionar múltiples contenedores y administrar la configuración de variables de entorno de manera efectiva.*

### *2. Gestión de Variables de Entorno con Docker*

- ***Descripción**: Utilizar la funcionalidad de variables de entorno en Docker para proporcionar configuraciones dinámicas a la aplicación Angular.*
- ***Consideraciones**:*
  - *Definir las variables de entorno relevantes en los archivos Dockerfile o en archivos de configuración específicos, utilizando diferentes valores para cada entorno (por ejemplo, dev, test, prod).*
  - *Utiliza comandos de Docker para establecer las variables de entorno al ejecutar un contenedor en un entorno específico.*

### *3. Uso de Imágenes de Docker Multi-etapa*

- ***Descripción**: Aplicar el enfoque de imágenes de Docker de múltiples etapas para optimizar el tamaño y rendimiento de las imágenes de contenedores.*
- ***Consideraciones**:*
  - *Configura diferentes etapas para construir la aplicación Angular en cada entorno, permitiendo así la optimización de la imagen final para su respectivo entorno.*
  - *Aprovecha la capacidad de las imágenes multi-etapa para incluir variables de entorno específicas en cada etapa, adaptando la construcción para entornos específicos.*

### *4. Orquestación de Contenedores con Kubernetes*

- ***Descripción**: Implementar una estrategia de orquestación de contenedores con Kubernetes para gestionar despliegues y escalabilidad en entornos de producción.*
- ***Consideraciones**:*
  - *Utiliza ConfigMaps en Kubernetes para gestionar la configuración y variables de entorno de la aplicación Angular en diferentes entornos.*
  - *Emplea Secrets en Kubernetes para manejar información confidencial, como claves de acceso a servicios externos, de manera segura.*

## Monitorización y Optimización del Rendimiento

Habla sobre la importancia de monitorizar y optimizar el rendimiento de las aplicaciones Angular. ¿Cómo identificarías cuellos de botella de rendimiento e implementarías optimizaciones?

### *Importancia de la Monitorización y Optimización del Rendimiento*

*La monitorización y optimización del rendimiento de las aplicaciones Angular son fundamentales para garantizar una experiencia fluida y eficiente para los usuarios, pudiendonos sacar provecho en los siguientes aspectos:*

- ***Experiencia del Usuario**: Un rendimiento óptimo garantiza una experiencia de usuario satisfactoria, incluyendo tiempos de carga rápidos y una navegación fluida.*
- ***SEO y Posicionamiento**: El rendimiento de la aplicación angular influye en el posicionamiento en buscadores, ya que los motores de búsqueda valoran la velocidad de carga y la experiencia del usuario.*
- ***Ahorro de Recursos**: Optimizar el rendimiento puede reducir el consumo de recursos del servidor y de ancho de banda, generando ahorros significativos.*
- ***Escalabilidad**: Identificar y optimizar cuellos de botella permite escalar la aplicación de manera más eficiente a medida que aumenta la demanda.*

### *Identificación de Cuellos de Botella de Rendimiento*

*Algunas estrategias para identificar cuellos de botella y llevar a cabo optimizaciones son:*

- ***Herramientas de Desarrollador del Navegador**: Utilizar las herramientas integradas en los navegadores para identificar cuellos de botella, como la pestaña de rendimiento en Chrome DevTools, que permite grabar y analizar el rendimiento de la aplicación.*
- ***Análisis de Carga Asíncrona**: Identifica las solicitudes de red y la carga de recursos para identificar posibles cuellos de botella relacionados con la red.*
- ***Uso Excesivo de Recursos del Cliente**: Monitoriza el rendimiento del lado del cliente para identificar operaciones costosas o ineficientes que ralentizan la interfaz de usuario.*

### *Implementación de Optimizaciones*

- *Utilizaria técnicas de compresión y minificación de archivos CSS, JavaScript y recursos estáticos para reducir el tamaño de descarga.*
- *Implementaria la carga diferida y asíncrona de recursos para cargar solo lo necesario inicialmente, optimizando el tiempo de carga inicial.*
- *Utilizaria formatos de imagen eficientes y técnicas de compresión para reducir los tamaños de archivo y mejorar los tiempos de carga.*
- *Emplearia estrategias de almacenamiento en caché y memorización de solicitudes para reducir la cantidad de peticiones al servidor.*

## Documentación del Proyecto

Explica la importancia de la documentación del proyecto en el desarrollo Angular. Describe los tipos de documentación que crearías y mantendrías a lo largo del ciclo de vida del proyecto.

*La documentación del proyecto en el desarrollo es de vital importancia para garantizar la comprensión, mantenibilidad y colaboración efectiva entre los miembros del equipo y otros*

*interesados.*
*A continuación describo los tipos de documentación clave que considero necesarios a lo largo del ciclo de vida del proyecto:*

***Tipos de Documentación a lo Largo del Ciclo de Vida del Proyecto:***

- ***Documentación de Arquitectura**:*
  - *Descripción de la arquitectura de la aplicación, incluyendo patrones de diseño, estructura de carpetas, y la interacción entre componentes.*
- ***Documentación Técnica**:*
  - *Documentación detallada de implementación técnica, como la configuración del entorno de desarrollo, la instalación de dependencias y buenas prácticas de codificación.*
- ***Documentación de API y Servicios**:*
  - *Descripción de las API y servicios utilizados en la aplicación, incluyendo la estructura de las solicitudes y respuestas, así como ejemplos de uso.*
- ***Documentación de Pruebas**:*
  - *Descripción de las pruebas unitarias, pruebas de integración y pruebas de extremo a extremo, incluyendo instrucciones para ejecutarlas y comprender sus resultados.*
- ***Documentación de Despliegue**:*
  - *Instrucciones detalladas para el despliegue de la aplicación en diferentes entornos, incluyendo configuraciones específicas y requisitos del sistema.*

## Manejo de Datos Asíncronos

Explica cómo manejarías problemas relacionados con operaciones asíncronas en Angular, como llamadas a API o eventos del usuario. ¿Cuál sería tu enfoque para garantizar la consistencia y la sincronización de datos?

***Respecto al manejo de llamadas a API Asíncronas me enfocaria en :***

- *Utilizar Observables de RxJS para manejar llamadas asíncronas a API ya que los Observables proporcionan una forma flexible de manejar múltiples operaciones asíncronas y de manejar secuencias de eventos.*
- *Implementar el manejo adecuado de errores mediante operadores como catchError para garantizar la robustez ante respuestas de API inesperadas.*
- *Usaria interceptores HTTP para centralizar la lógica de manejo de respuestas y solicitudes HTTP, esto permite transformar datos de manera consistente y manejar errores de forma global.*

***Respecto al manejo de Eventos del Usuario Asíncronos:***

- *Utilizaria directivas de eventos para capturar e interpretar las interacciones del usuario, como clics, desplazamientos, etc. Esto permite manejar eventos de manera específica y consistente en toda la aplicación.*

- *Implementaria técnicas basadas en Observables y Subject de RxJS para actualizar la interfaz de usuario de manera reactiva en respuesta a eventos del usuario.*

***Garantizaria la Consistencia y Sincronización de Datos:***

- *Utilizando herramientas como NgRx para mantener un estado centralizado de la aplicación, lo que facilita la sincronización y la consistencia de los datos en toda la aplicación.*
- *Implementaria patrones de transacciones atómicas para garantizar la consistencia de los datos en operaciones complejas que involucren múltiples modificaciones asíncronas.*

## Migración de Versiones

¿Que consideraciones tendrías en cuenta al migrar de una versión anterior a una más reciente de Angular? Describe tus pasos para minimizar los problemas durante el proceso de migración.

***Tendria presente las siguientes consideraciones:***

- *Comprender los cambios significativos en la API de Angular, así como las versiones en desuso o eliminadas para garantizar la compatibilidad con tu código existente al migrar a la nueva versión.*
- *Verificar la compatibilidad de las dependencias del proyecto, como las bibliotecas de terceros y herramientas relacionadas, con la versión de Angular a la que se planea migrar.*
- *Asegúrarse de que la nueva versión de Angular sea compatible con los navegadores web utilizados por tu aplicación, teniendo en cuenta el soporte de navegadores que ofrece la nueva versión.*
- *Planificar pruebas para validar que las funcionalidades clave de la aplicación sigan funcionando según lo esperado después de la migración.*

***Pasos para Minimizar Problemas durante la Migración:***

1. ***Análisis de Cambios y Notas de Versión**:*
   - *Investigar las notas de versión y documentación oficial de Angular para comprender los cambios, novedades y consideraciones específicas al migrar a la nueva versión.*
2. ***Actualización Gradual**:*
   - *Si la distancia entre la versión actual y la versión objetivo es significativa, consideraria realizar actualizaciones incrementales a versiones intermedias para minimizar la complejidad y los problemas potenciales.*
3. ***Ejecución de Herramientas de Actualización**:*
   - *Utilizar las herramientas de actualización proporcionadas por Angular, como Angular Update Guide y Angular CLI tools, para detectar y aplicar automáticamente cambios necesarios en el código, como modificaciones de API y ajustes en la estructura del proyecto.*

4. ***Pruebas Automatizadas**:*
    - *Desarrollar pruebas automatizadas que cubran las funcionalidades críticas de la aplicación antes de la migración, y ejecútalas nuevamente tras la migración para identificar posibles problemas.*
5. ***Refactorización Basada en Linter y Analizadores Estáticos**:*
    - *Utilizar herramientas de análisis estático, como linters y analizadores de código estático para identificar y abordar problemas de compatibilidad, obsolescencia y buenas prácticas durante el proceso de migración.*
6. ***Actualización de Dependencias Adicionales**:*
    - *Asegúrase de actualizar todas las dependencias y bibliotecas relacionadas, como Angular Material o módulos complementarios, para que sean compatibles con la nueva versión de Angular.*
7. ***Entorno de Desarrollo Aislado**:*
    - *Considerar realizar la migración en un entorno aislado o en una rama separada del repositorio para minimizar el impacto en el desarrollo y las pruebas del flujo de trabajo principal.*

## Resiliencia y Manejo de Excepciones

¿Cómo garantizarías la resiliencia de una aplicación Angular frente a errores y excepciones inesperadas? Habla sobre técnicas para manejar errores de manera elegante y minimizar el impacto en la experiencia del usuario.

*Algunas estrategias para manejar errores de manera elegante y minimizar su impacto en la experiencia del usuario que yo usaria serian:*

- *Utilizar interceptores HTTP para capturar y manejar errores de red y solicitudes HTTP, lo que permite tomar medidas específicas, como mostrar mensajes de error significativos o redireccionar a páginas de error personalizadas.*
- *Crear componentes reutilizables para notificar al usuario sobre errores importantes, como alertas emergentes o banners de error, de manera que la retroalimentación sea clara y visible.*
- *Al realizar operaciones críticas, como eliminación de datos, presenta diálogos de confirmación que soliciten validación por parte del usuario para minimizar la posibilidad de errores accidentales.*
- *Diseñar y planificar estados de carga y error en la interfaz de usuario para ofrecer una experiencia fluida incluso cuando ocurran errores inesperados.*

- *Implementar estrategias de recuperación, como la recarga automática de datos o la reanudación de operaciones interrumpidas, para minimizar el impacto de los errores en la experiencia del usuario.*

- *Utilizar operadores de RxJS como catchError para manejar excepciones en observables, y garantiza que las promesas estén correctamente manejadas mediante el manejo de errores en las funciones then y catch.*

## Autoaprendizaje y enseñanza

¿Cómo te mantienes motivado y enfocado durante el autoaprendizaje?
¿Cómo integrarías el autoaprendizaje en tu rutina diaria como desarrollador de software?
Mencione una librería de angular que permita facilitar su trabajo como desarrollador y además en pocas palabras explique su funcionamiento e implementación.

- *Me mantego motivado poniendo en practica una organizacion y planificacion clara en mis actividades, logrando asi micro logros a diario lo que permite ver de cierta manera un progreso diario.*

- *Integro el autoaprendizaje en mi rutina diaria reservando para ello un tiempo especifico cada dia, escogiendo este tiempo estrategicamente como por ejemplo al inicio de la jornada, adicionalmente intento practicar regularmente desarrollando proyecto propios.*

- *RxJS es una librería de programación reactiva que amplía las capacidades de Angular al permitir la composición y el manejo elegante de secuencias de eventos asíncronos, como operaciones de red, interacciones de usuario y otros eventos. Se basa en Observables y ofrece numerosos operadores para filtrar, transformar y combinar datos reactivamente. RxJS me ayuda a manejar de manera eficiente la lógica asíncrona y a construir aplicaciones más reactivas y eficaces.*

# Ejercicio Práctico: Aplicación de Cat Breeds

## Descripción:

Desarrolla una aplicación web utilizando Angular que permita al usuario seleccionar una raza de gato de una lista desplegable y visualizar información relevante sobre esa raza, incluida una imagen. Además, la aplicación debe proporcionar un campo de búsqueda para encontrar razas de gatos por nombre.

## Requisitos Funcionales

Yo como usuario requiero una plataforma que conectada a la API de The Cat Breeds

https://developers.thecatapi.com/view-account/ylX4blBYT9FaoVd6OhvR?report=FJkYOq9tW

Me permita tener las siguientes funcionalidades:

Tener una vista donde pueda ver en resumen las **razas** de gatos con una imagen y una pequeña descripción esta vista debe estar tabulada y solo cargar 10 razas iniciales además de un botón de ver más que al dar clic me carguen 10 razas adicionales el botón debe persistir hasta que no queden más razas para cargar.

La vista anterior debe contener un filtro de tipo texto que al escribir vaya filtrando las razas de gatos de acuerdo al texto escrito.

Cada tarjeta de razas del punto anterior debe contener un botón de detalle que me lleva a un módulo donde se me presenta las diferentes imágenes de la raza y toda la información asociada a esta, además de un link que me lleva a su descripción en Wikipedia en una nueva pestaña.

## Requisitos Técnicos:

La aplicación debe desarrollarse utilizando el framework Angular.

## Servicio de Datos:

Utiliza un servicio de Backend o una API pública
(TheCatAPI) para obtener información sobre las razas de gatos y sus imágenes.

## Componentes Angular:

Utiliza componentes y librerías de Angular para modularizar y organizar tu aplicación de manera efectiva.

## Formularios y Eventos:

Utiliza formularios y gestiona eventos para la interacción del usuario, como la selección de una raza de gato y la búsqueda de razas.

## Estilo y Diseño:

Aplica estilos CSS y funciones o librerías para mejorar la apariencia y la usabilidad de la aplicación.

## Entregables:

Código fuente cargado en git de la aplicación Angular. Instrucciones de como ejecutar la aplicación localmente.