# Introducción

## About Me

Mi nombre es Starling Germosen, bueno, Ingeniero Germosen, como suelen poner los possers en sus facebooks cuando se gradúan, no hagan eso, eso le resta mucho a su perfil profesional, lo importante en estos tiempos no es tu título, sino lo que tú has realizado o hasta dónde has llegado y más en este mundo de la tecnología.

Soy Master en Comercio Electrónico con titulación de la universidad APEC, fundador de Juegos Mentales Online, una plataforma que genera dinero compartiendo retos y desafíos para los perfiles sociales de ciertas empresas, con clientes como Malta Morena y Perfumuda y más.

CEO de MersyRD, una solución de expedientes médicos para clínicas, hospitales y doctores independientes.

Creador de Torneo Predicciones, una app móvil desarrollada en Xamarin con el fin de desafiar a tus amigos intentando acertar predicciones y ganar puntos en el proceso.

Entre otras aplicaciones que pueden auditar en GitHub.

Analista programador en Dirección General de Impuestos Internos.

Además de tener 12 años de experiencia en el desarrollo a la medida de soluciones informáticas, páginas web y consultoría para diversas instituciones y pequeñas empresas en PraySoft Dominicana.

## Que Vamos a Ver

En este curso vamos a trabajar con ASP .Net Core, a diferencia de lo que se había planteado originalmente, claro, esto es si ustedes así lo desean, puesto que las diferencias entre .net core y MVC, son tan ínfimas, que no tiene sentido enseñarles algo que ya tiene 2 años de desfase, además de que los conocimientos adquiridos en .netCore, ustedes lo pueden llevar al mundo MVC sin problemas, con la ventaja de que si desarrollan directo en .netCore, pueden tener un desarrollo multi-plataforma.

La única diferencia radical que tendríamos entre .netCore y MVC, serían las inyecciones de dependencias y las configuraciones, cuando lleguemos a ese tópico en los módulos finales, me comprometo a enseñarles cómo se hacen las configuraciones en ambos y las inyecciones de dependencia, ya que cuando ustedes salgan a la calle se van a topar con muchas aplicaciones hechas en MVC.

La orientación de este curso es enfocada a convertirnos en Fullstack developer o desarrollador completo, pues hay que ser realistas, en el mundo real las empresas pequeñas con las cuales usted puede conseguir una picota, no andan contratando un desarrollador para el Front-End y otro para el Back-End, normalmente quieren uno que meta mano en todo, si bien es cierto que es bueno especializarse, la idea de este curso es que usted salga de acá a meter mano y a ganar dinero, por lo tanto, aunque usted no se debería volver un master en diseño, al menos deberías conocer:

Cuales reglas básicas de usabilidad debe seguir una aplicación.

Cuáles Frameworks existen para facilitar tu trabajo y hacerlo ver como algo profesional.

Y eso no es solo una mentalidad tercermundista de RD, eso pasa en todos lados del mundo, solo las grandes empresas contratan gente especializada y normalmente esas solo sub contratan el trabajo.

Pero, aun cuando usted caiga en un lugar que contrate gente especializada, imagine que usted no sepa nada de lo que es Css? Una cosa es que usted no tenga la vena de mariconcito, pero si usted no sabe lo básico de estilo, no se va a entender nunca con el diseñador, o con el encargado de infraestructura, por eso la orientación del curso será a meter mano con todo lo que involucra la creación de un proyecto completo desde cero.

Ojo, este no es un curso de programación ni de C Sharp, a pesar de que la forma en que yo explico es muy básica y de hecho es la que me ha funcionado con mis estudiantes de clases particulares, es bueno que conozca al menos de manera básica lo que es C# y conocimientos medios de programación orientada a objetos.

Algunos de los tópicos que vamos a tocar son:

* Construyendo interfaces de usuario.
* Contrayendo APIs
* Programación Orientada a Objetos de verdad, verdad.
* Arquitectura Limpia
* Manipulación de datos con Entity Framework Core
* Manipulación de objetos con Linq
* Servicios de Azure

Todo esto y muchos conceptos más, los iremos viendo y aprendiendo en el contexto de una aplicación del mundo real, partiendo desde un documento de requerimientos, extrayendo los casos de uso y el análisis de cómo implementar estos.

## Prerrequisitos

* Conocimientos básicos de C#,
* Conocimientos básicos de POO,
* Conocimientos de modelado y estructura de datos.
* Visual Studio Community Igual o Superior al 2017
* Net Core Mayor o igual a 2.2

Nosotros no vamos a profundizar en aquellas herramientas que no tengan una implicación directa con la creación de nuestro programa, pero debemos conocer la mayoría de ellas al menos por arribita, corresponderá a la tutoría personalizada la parte de profundización sobre algún tema x que no tenga que ver con el desarrollo de nuestra aplicación o investigarlo por su cuenta, sin embargo siéntase libre de preguntar que yo no muerdo y soy muy abierto a los comentarios y debates, si hay algo de la clase que no les gusta lo podemos cambiar o algo que se necesite reforzar lo podemos debatir.

Nuestra aplicación, a pesar de ser un app de consumo, asumo que ustedes conocen la diferencia entre aplicaciones de consumo y empresariales, la vamos a enfocar a cómo funciona el mercado en la calle, el cual es bastante distinto a como nos enseñaron en la universidad, nuestro proyecto final, será una especie de mini red social para coaching donde los coach, pueden subir sus eventos y la gente puede darle seguimiento, una especie de EventBride pero orientado al nicho de mercado del coaching a ver si nos lo compra Herbalife o Amway, así que no vayan a creer que haremos una aplicación que le haga competencia a Facebook, aunque los conocimientos que vamos a adquirir aquí, si son suficientes para hacer una red social en .net completa, pero no es el objetivo del curso, háganla ustedes si así lo desean, de igual forma los conocimientos y el modo de trabajo que vamos a emplear, te va a servir para cualquier tipo de aplicación web que te puedas imaginar.

En este curso te llevaremos de la mano sin importar si eres un desarrollador junior o novato que apenas está incursionando en el mundo de la programación, iremos desde lo más básico hasta lo más avanzado para que mejores tus habilidades de desarrollo.

Iniciaremos conociendo la plantilla básica de netCore, como está estructurada y cómo funciona el esqueleto, para posteriormente ir transformándola en nuestra aplicación.

La aplicación como mencione previamente es una red social para conectar a coachs con personas que les interesan esos temas, los coach, pueden registrar sus eventos próximos, los usuarios pueden ver todos los eventos e indicar que van a asistir a estos, así como seguir al coach para que se les notifique de la creación o cancelación de algún evento futuro, las personas también pueden buscar eventos por coach, tipo de charla o ubicación, los eventos que el usuario marca se añaden al calendario.

## ¿Qué aprenderemos?

Para desarrollar nuestra aplicación veremos algunos elementos del desarrollo web moderno, tales como

Bootstrap, Css, Usabilidad, Javascript, Ef code fisrt, RESTful APIs, seguridad, Plain Old CLR Object, object-oriented design, y lo que vallamos necesitando en el camino según lo requiera la trama.

Nosotros vamos a iniciar con tópicos básicos, quiero que esa parte quede bien clara, por tanto el código inicial no estará optimizado y será la vergüenza de los dioxes del software, pero a medida que vayamos avanzando iremos evolucionando el mismo a uno más limpio, mantenible y elegante, que hable por sí mismo con solo verlo sin necesidad de tener que añadir comentarios adicionales, práctica que se conoce como código suficientemente descriptivo o auto descriptivo.

Te garantizo que con esta guía, vas a poder llevar tus habilidades en el desarrollo .net al próximo nivel,

## Estructura del Curso

La estructura del curso será impartida o se divide en tres partes, fundamentos, tópicos avanzados, y finalmente arquitectura.

## Fundamentos

En la primera parte abarcaremos los tópicos tales como, ¿por dónde arranco?

Por eso en esta parte iniciaremos con el documento de requerimientos, te enseñare a extraer los principales casos de uso, que tienen un mayor impacto en el desarrollo de nuestra aplicación y luego la construiremos paso a paso, el enfoque de esta parte es enseñarte la mentalidad del ingeniero de software, pensar como lo haría un ingeniero de software, como abordar un problema romperlo en problemas más pequeños y resolverlos paso a paso. De igual forma tocaremos los cimientos del desarrollo de software Backend and Front End, usando y aprendiendo en el camino.

Entity Framework Code First, Bootstrap, validaciones de datos, seguridad, aspectos artísticos de interfaz de usuario para crear interfaces amigables, etc.

## Tópicos Avanzados

En la segunda parte, implementaremos un servicio de notificaciones.

En esta parte haremos bastante énfasis en el diseño orientado a objetos, construir APIs y valernos de Botstrap para construir aplicaciones modernas.

## Arquitectura

Y finalmente en la última parte veremos un poco de arquitectura.

## Vamos a iniciar!!!!

Te recomiendo que aunque creas que sabes demasiado de lo básico, no te saltes directo a la siguiente parte, porque créeme que hay algunos truquitos que quizá no conozcas y que te serán de mucha ayuda en tu desarrollo profesional, en la primera parte es donde explicamos todos los conceptos, por lo que en la segunda y tercera parte ya nos vamos a enfocar directamente en trabajar sin explicar mucho a no ser que sea algo nuevo que no hayamos implementado antes.

Otra cosa de vital importancia para el aprendizaje, no importa que tan básico consideres que sean los ejercicios, te recomiendo encarecidamente que intentes resolverlos por tu cuenta, sin ver la solución primero, intenta por tu cuenta y después consulta la respuesta, también toma en cuenta que no existe una sola forma de hacer las cosas, puede que algunas de las respuestas que yo de a algún ejercicio no este 100% optimizada por el nivel en el que nos encontremos por lo que quizá el tuyo este mucho mejor, en programación, a pesar de que todo es cero y uno, un resultado se puede conseguir por diversos medios, uno más óptimos que otros, pero mientras estamos aprendiendo lo importante es hacerlo.

## Que es .Net Core

ASP.NET Core es un nuevo Framework de código abierto y multiplataforma para la creación de aplicaciones modernas conectadas a Internet, como aplicaciones web y APIs Web, aunque por ahí viene también las aplicaciones Winforms, pero eso está en veremos.

Se diseñó para proporcionar un Framework de desarrollo optimizado para las aplicaciones que se implementan tanto en la nube como en servidores dedicados en las instalaciones del cliente.

Se pueden desarrollar y ejecutar aplicaciones ASP.NET Core en Windows, Mac y Linux.

ASP.NET Core puede ejecutarse sobre el Framework .NET completo o sobre .NET Core.

.NET Core es una nueva versión modular del Framework .NET que permite el uso multiplataforma de .NET. Es un subconjunto del Framework .NET por lo que no tiene toda la funcionalidad del Framework completo, y puede emplearse para creación de aplicaciones web, de escritorio y móviles.

El uso del Framework completo nos permitirá poder añadir cualquier dependencia que necesitemos del Framework, pero perderemos todas las ventajas que tienen las aplicaciones .NET Core, tales como la multiplataforma, la mejora del rendimiento, el menor tamaño de las aplicaciones, etc.

**Por qué utilizar ASP.NET Core en lugar de ASP.NET?**

ASP.NET Core es un rediseño completo de ASP.NET. No es una actualización de ASP.NET 4, por lo que su arquitectura ha sido diseñada para resultar más ligera y modular.

ASP.NET Core no está basado en System.Web.dll que aportaba un exceso de funcionalidad. Se basa en un conjunto de paquetes NuGet granulares y bien factorizados. Esto te permite optimizar tu aplicación para incluir solo los paquetes NuGet que necesitas.

**Beneficios de ASP.NET Core contra ASP.NET**

Seguridad más estricta: Menor intercambio de información y rendimiento mejorado, ya que está formado por paquetes NuGet, lo que permite una modularidad total, de ésta forma solo añadiremos los paquetes con la funcionalidad que necesitemos.

Una plataforma unificada para la creación de interfaz web y las APIs web.

Integración de los Frameworks modernos de cliente y flujos de trabajo de desarrollo.

Un sistema de configuración basado en la nube. Preparado para su integración de forma sencilla en entornos en la nube.

Inyección de dependencias incorporada.

Las peticiones HTTP se procesan siguiendo un flujo que puede ser modificado de forma modular para adaptarse a nuestras necesidades y que nos permite poder controlar el procesado de las peticiones HTTP en nuestra aplicación.

Capacidad para alojar en IIS u otros servidores web como Apache. o self-host en su propio proceso.

Nuevas herramientas que simplifican el desarrollo web moderno.

Crea y ejecuta aplicaciones multiplataforma ASP.NET Core en Windows, Mac y Linux.

De código abierto y orientado a la comunidad.

## Sistema de Control de Versiones

Sistemas de control de versiones, la idea de estos es que tu tengas tu proyecto en el servidor siempre resguardado de cualquier imprevisto, cuando vayas a trabajar algún cambio, la idea es que bajes los cambios más recientes a una carpeta de trabajo en tu computadora, cada vez que tienes algo que funciona o completa una responsabilidad que se te haya delegado en el equipo, tú haces un commit al repositorio, para que los cambios que están en tu maquina se combinen con los que están en el servidor, el repositorio no es más que una mini base de datos, que guarda el historial de todo tipo de cambios que se hayan hecho al código o los archivos que sean parte del proyecto. Los beneficios de trabajar siempre ordenados y con un control de versiones es que podemos devolvernos en cualquier momento a cualquier estado de la aplicación, como nuestra máquina del tiempo con más coherencia que las líneas temporales de avengers. Los controles de versiones nos sirven para comparar elementos o cambios que hayan realizado otros compañeros del equipo, descartar cambios entre otras funcionalidades chulas que iremos abarcando más adelante. Existen múltiples controles de versiones, están tfs, SourceSafe para la gente que desarrollaba para entornos Microsoft, scm, SubVersion que fue uno de los más famositos en el mundo anti microsoft, hasta que Linus Torvalds creó Git en 2005, el cual ha reemplazado a todos los demás, no confundir sistema de control de versiones con programa de control de versiones, el sistema es la forma en que funciona algo de manera conceptual, el programa es el que permite llevar esa conceptualización al mundo de la informática.

Dentro de los programas para control de versiones con la metodología Git, podemos encontrar GitLab, que tiene una versión en línea como descargable para montar en tu propio servidor, Team Fundation Server o Azure DevOps, y GitHub.

## GitHub

Vamos a conocer al Octu-Cat, el gato pulpo, github es una plataforma colaborativa que permite compartir nuestro código fuente con la comunidad de desarrolladores y sirve a su vez para hacer BAM, así como que las empresas puedan ver tus proyectos, como trabajas, como codeas y de esa forma sumes puntos en la comunidad de desarrollo, también github te permite mantener el traking del código de tu aplicación, ya sea de manera privada o pública, se vende como la red social de los desarrolladores, pero nunca fue así, aunque haga muchos esfuerzos en convertirla en tal. Es propiedad de Microsoft en la actualidad.

Github o cualquier otro control de versiones nos evita el tema de estar guardando en el disco duro nuestros códigos fuentes y que de repente pase algo que haga que se pierda, o estar con el can de guardar proyecto uno, proyecto uno final, proyecto uno final este si o final final.

Vamos a ir a GitHub.Com, Creamos un usuario, Creamos un Repositorio sea público o privado. Dentro del Repositorio creamos un proyecto llamado ThuRealProyecto. Añadimos columnas donde guardaremos el estado de las tarjetas o asignaciones, estas son para poder tener una visión del estado de nuestro proyecto.

Creamos Pendiente, escogemos la plantilla de ToDo escogemos como se van a trabajar las nuevas tarjetas, en mi caso voy a especificar que desde que yo creo una venga a Pending, porque como soy un único desarrollador, si yo mismo fui quien la creo, es porque las voy a trabajar, en una organización las cosas no funcionan así, pues el planning lo hace otro, pero en nuestro caso, lo vamos a trabajar de esta forma. De igual manera si por alguna razón se re abre algún pendiente, quiero que se mueva acá.

Cuando nosotros trabajamos en ambientes colaborativos, se da algo que se llama Pull Request, eso lo veremos más adelante, por lo pronto, ignoraremos estos dos apartados, pues yo no quiero que me llenen de posibles basura mi proyecto.

Luego procedemos a crear la Columna Working donde vamos a mover las actividades cuando las estemos trabajando. Escogeremos la plantilla In Progress, acá no modificaremos nada más, podríamos escoger que se muevan acá las actividades que sean revisadas, pero eso no es práctico, pues lo correcto es que el desarrollador especifique cuando inicio a trabajar en algo, no que me digan wey pana, pero tú tienes 4 días atrasado en tal tarea y tu estés aéreo al respecto porque estabas trabajando en otro proyecto.

Y creamos la última columna que llamaremos Finalizado donde escogeremos la plantilla Done, seleccionamos que se muevan aquí las tareas cuando sean cerradas y no haremos nada con las que vengan de Pull Request

Podemos añadir más columnas según sea necesario, pero con esas son suficientes para nuestro objetivo.

Posterior a eso vamos a proceder a descargar GitHub Desktop e instalarlo, también podemos instalar la extensión de github para visual studio por la opción de tools, extentions si no queremos usar programas adicionales, yo prefiero usar la versión de escritorio que me sirve de paso para otros controles de versiones como GitLab, el cual antes me permitía par de chulerías que GitHub no tenia, pero que ahora mismo aunque gitlab tiene muchas más herramientas que github yo no uso más que las ramas, los estados de trabajo y las tareas y eso ya se puede hacer por github, pero igual prefiero el programa.

## File New Project por fin!!!!

Vamos ahora a crear nuestro nuevo proyecto, primero que nada vamos a clonar nuestro repositorio en un folder cualquiera de nuestro disco duro.

Procedemos a crear el nuevo proyecto dentro del folder.

Una vez creado lo que vamos a hacer es sincronizar los cambios de lo que tenemos local con lo que está en el servidor, para evitar que se nos rompa el código y no podamos devolvernos en caso de hacer un disparate, los commits se deben hacer cada vez que tengamos un mínimo de funcionalidad resuelta.

Los archivos que podemos ver con el signo de + en verde son archivos que se están añadiendo desde el ultimo commit o más bien si lo comparamos con la rama en la que nos encontramos para lo que está en el server, es decir, ese archivo esta en nuestra carpeta pero no está en el servidor.

Los comentarios son etiquetas que nos permiten entender en el futuro que estábamos trabajando en el momento en que decidimos crear este commit.

Una de las herramientas que usaremos de manera opcional es

reSharper, la cual pueden conseguir con una licencia de estudiante gratis por un año y esta te ayudara bastante en la optimización de código, pero sin embargo esta herramienta no es imprescindible para nada y más ahora que code lense es gratuito con visual studio comunity 2019

También vamos a usar una extensión gratuita, aunque quizá no la vamos a fondo, llamada web essentials, la cual nos permite hacer modificaciones en vivo en el html del proyecto los cuales se reflejan de forma automática en el código de nuestro proyecto, es decir, nos permite hacer un link entre lo que esta en el navegador y el proyecto que estamos depurando y hacer ediciones directo en el navegador para que modifiquen el archivo.

Para eso vamos a extensiones y actualizaciones nos vamos a online y escribimos web essentials

También de manera opcional Productivity Power Tool

## Resumen

Vamos ahora a ver lo que aprendimos en esta lección introductoria,

Conocimos al profesor,

Vimos cual será la forma de trabajo y los conocimientos que vamos a adquirir al finalizar el curso, así como las herramientas que vamos a implementar y el proyecto que trabajaremos.

Aprendimos lo básico acerca de .NetCore y las diferencias entre él y su predecesor, de igual forma vimos las diferentes plantillas que hay disponible de manera predeterminada y las diferencias entre estas.

Aprendimos sobre lo que es un control de versiones, su importancia y comenzamos a usar un programa llamado github que nos permite trabajar con esta metodología.

# Extrayendo los principales casos de uso del requerimiento.

Una de las preguntas que más se hacen los desarrolladores cuando están iniciando es ¿Por dónde empiezo? Normalmente siempre arrancamos por el código, lo cual es un error garrafal,

Es por eso que es lo primero que vamos a cubrir en esta sección

Quiero que nos enfoquemos en un documento de requerimientos

Y luego vamos a ver los siguientes pasos que debemos de dar

Vamos a extraer los principales casos de uso

Que son los que tendrán mayor impacto en el diseño de tu aplicación, en lugar de implementar cada característica de principio a fin y totalmente pulido, nosotros vamos a implementar estos casos principales en primer lugar, porque implementar primero estos casos son los que nos darán una idea de los desafíos venideros y son los que nos permitirán tener un prototipo medianamente funcional, así como ver los principales retos involucrados con nuestro proyecto ahora echemos un vistazo al documento de requerimientos.

Típicamente el desarrollo de software inicia con un documento de requerimientos.

A veces el documento puede ser una pequeña página del tamaño de un post it o puede ser un extenso documento completamente detallado de 500 páginas, (que nadie va a leer), independientemente de cómo te sea entregado estos, tu labor como ingeniero de software es extraer cuales son los caos de uso con los cuales puede trabajar un desarrollador o en este particular tú, en este caso veremos el documento de requisitos para nuestro proyecto.

“Coaching events es una mini red social que permite a coachs comunicarse con sus seguidores para que estos participen en sus conferencias, permite crear y listar sus charlas, las cuales para ser creadas requieren una fecha, tipo, y la ubicación.

Un coach debe tener una página donde pueda editar o eliminar o añadir un evento a la lista.

Los seguidores y amantes de las conferencias pueden seguir a sus coachs favoritos y recibir notificaciones cuando ocurra un cambio en algún evento.

Los usuarios deben tener la posibilidad de ver los próximos eventos o buscarlos ya sea por coach, tipo o ubicación así como ver el calendario en detalle y añadirlo a su propio calendario de actividades.

Cuando el usuario sigue a sus coach favoritos ellos deben poder ver en su pantalla de inicio sus próximos eventos.”

Este es básicamente un documento de requerimientos, puede ser más extenso o perfeccionado pero con este podemos ir metiendo mano por ahora, no podemos desperdiciar mucho tiempo en esto, después de todo normalmente este documento nos lo dan a nosotros, no lo redactamos, lo verdaderamente importante para el desarrollador es saber extraer sus casos de uso.

Estos se expresan con pocas palabras, por ejemplo en el primer párrafo

Podemos identificar dos casos de uso

Crear una cuenta y crear un evento.

Expresar en pocas palabras los casos de uso ayuda a un mayor entendimiento del proceso y simplifica la comunicación así como reducir los problemas de interpretación que traen consigo esos documentos extensos con un montón de detalles, los detalles pueden venir después cuando nos acerquemos a la fase de implementación, vamos ahora a tomar unos minutos a extraer todos los casos de uso.

Y en breve compararemos tu solución con la mía.

## Autenticación

Módulo de registro, módulo de acceso, opción para cerrar sesión, módulo de cambio de contraseña, edición de perfil.

Acá podemos ver algo interesante, a pesar de que en el documento de requerimientos no se me dijo que debía crear una pantalla de acceso, es obvio que hace parte de nuestro proyecto, por lo que muchas veces como ingenieros, corresponde a nosotros agregar esos casos o pasos, sin los cuales no podemos realizar los procesos que se nos están pidiendo, y por eso es que debemos valernos de nuestra experiencia para que al momento de que un emprendedor o empresa nos venga con esos requerimientos ambiguos, nosotros podamos cobrar lo justo por nuestro trabajo, ya que al tener experiencia sabemos todas las aristas adicionales que debemos emplear para poder cumplir con los requerimientos

Por esto, los documentos de requerimientos, deben ser vistos como un entendimiento de alto nivel acerca de que es tu proyecto a nivel macro.

A medida que vayas avanzando, vas a usar tu propio criterio o mantenerte en constante comunicación con el cliente a medida que vas construyendo tu sistema para llenar esos vacíos existenciales y dudas que tengas en este caso use mi criterio para determinar que si tengo un registro, debo tener un login, log out, etc,

La buena noticia con respecto a la primera parte de los casos de uso, es que la plantilla por defecto de visual studio asp viene con una forma simple de construir tu sistema de autenticación y registro, si tu deseas puedes construir el tuyo propio, pero, ¿en verdad vale el esfuerzo extra que le vas a poner? ¿En verdad crees que vas a construir algo mejor que Microsoft estando al nivel en el que estas de desarrollo? Si la respuesta es no, perfecto, usa el que viene por defecto y enfoquémonos en cosas más importantes, puesto que hasta en el mundo real la mayoría de las grandes empresas están delegando la seguridad en sistemas de autenticación externos, ya muchas apps permiten la autenticación con Facebook o twitter, por poner un ejemplo, así que saca de tu mente la necesidad de querer hacer todo desde cero.

## Eventos

En la parte de eventos, he podido sacar lo siguiente: añadir un evento, listar mis próximos eventos, editar y eliminar un evento, ver todos los eventos próximos, Buscar eventos por distintos filtros, ver los detalles de un evento.

## Calendario

Añadir eventos a un calendario, remover eventos de tu calendario, ver los eventos a los que marque que voy a asistir

A pesar de que el tema de añadir los eventos no eran parte de los requerimientos en si nuestra experiencia como desarrolladores siempre deben permitirnos agregar valor a las propuestas ambiguas que tengan nuestros clientes, eso es lo que marca la diferencia entre un profesional y un pica pollero, sin denigrar a los que practican esta labor.

Tu experiencia debe ayudarte a llenar esos vacíos que el cliente no especificó, pero que tú sabes que él va a necesitar. Cuando estas en equipos de trabajo grandes, donde tú no eres otra cosa que una pieza más del equipo, no es muy bueno ponerse más creativo de la cuenta, en especial en latino américa, donde el jefe puede pensar que tú le quieres quitar el puesto o que ser más eficiente que el promedio puede hacer que el equipo no funcione de la manera correcta por la falta de sinergia, pero cuando estas en la calle bandeándotela como puedas, debes siempre dar lo mejor de ti,

## Seguimiento

Seguir un coach, dejar de seguir un coach, ver a quien estoy siguiendo, ver los eventos en mi feed de a quienes estoy siguiendo

De nuevo acá vemos cosas que no estaban en el documento de requerimientos original, pero como somos super analistas nosotros pudimos agregar estos.

Vamos ahora a Github y vamos a añadir todos estos ítems a nuestro proyecto. Yo solo voy a agregar dos, ahorita agrego los demás,

Tenemos dos formas de registrar, podemos irnos directamente a Issues o problemáticas, y registrar cada una de estas actividades como una problemática, pero esta opción es más viable cuando ya tenemos un sistema en producción, y recibimos colaboración de otros, en mi caso voy a ir al proyecto me situó en la pestaña o columna Pending, le damos a añadir y ponemos la nota que deseamos, desde acá al momento de programarlas las convertiré en problemática a resolver en el momento en que toque trabajarla, las problemáticas no son solo problemas de aplicativo, pueden ser documentación, solicitud de ayuda, preguntas, etc.

## Dependencias

Una vea hayamos extraído los casos de uso, toca verificar las dependencias e introducirlas en nuestra agenda de trabajo.

Por poner un caso, volviendo al slide anterior, no podemos dejar de seguir un coach si primero no lo hemos seguido, ¿Cierto?

No sé si recuerdan de la universidad en investigación de operaciones la famosa ruta crítica, creo que fue en esa materia, bueh, es un concepto similar el que se usa para determinar el orden de dependencia. Otro ejemplo para poder seguir a un coach primero debemos ver sus eventos porque al momento de ver los detalles de los eventos es donde vamos a mostrar al coach y es ahí donde el usuario podrá seguirlo, este orden no tiene que ser así, pero es la forma en que lo encuentro más lógico.

Las dependencias se representan con una flecha punteada señalando al padre de la funcionalidad de la que se depende.

Lo que quiere decir que por ejemplo, tenemos que implementar primero la función de ver eventos, Luego seguir un coach y finalmente para cerrar el ciclo, dejar de seguir.

Ahora la tarea que te toca es ver cuáles son las dependencias de trabajo entre las demás casos que planteamos exceptuando el de la parte de autenticación, solo de las relacionadas con el core de la aplicación.

Mantenlo simple, imagina o intenta lograr una segregación en la que cada caso no tenga más de una dependencia o dos como máximo.

Una vez que tenemos todas las dependencias establecidas, debemos iniciar por aquel que no tiene ninguna dependencia, pero que dé el dependen muchos más. De esa forma podremos tener los órdenes de ejecución de nuestro proyecto.

Vamos a tomarnos unos minututos para hacer esta tareita, en lo que yo las voy agregando todas al github.

Acá te muestro mi solución

En primer lugar pondré añadir un evento, si el coach no tiene la habilidad de añadir un evento, no hay nada que se pueda hacer en el sistema, nadie puede buscar, nadie puede añadir a favorito etc, Una vez implementado este, podríamos implementar la opción de ver los próximos eventos Y a partir de ahí podríamos editar o remover eventos, De igual forma una vez que tenemos la implementación de añadir eventos, podemos ver todos los eventos próximos, no solo los míos Y cuando tenemos este podríamos trabajar en añadirlos al calendario.

Posterior a eso podemos crear una página donde ver aquellos eventos a los que estoy asistiendo y a su vez removerlos del calendario. Volviendo a la parte donde tenemos implementado todos los eventos, podemos ir a los detalles y de esa forma seguir un coach así que consecuentemente podríamos crear una página para los usuarios para ver quienes está siguiendo este y en esa página tener la posibilidad de dejar de seguir un coach además cuando sigue un coach puede ver sus eventos en su propio feed y también cuando tenemos la forma de ver el listado de todos los eventos podemos implementar también la búsqueda de eventos o ver el detalle de un evento en concreto.

Si tu solución se parece a la mía, es genial, pero sino, no te preocupes, existen muchas formas de asimilar el todo de un proyecto, así que no te preocupes, quizá la solución tuya es mucho mejor que la mía de hecho.

Ahora es momento de identificar nuestra ruta crítica estos casos de uso principales son los que le dan forma al dominio de nuestra aplicación este es el que involucra la captura de data y cambia el estado de la aplicación así que de esta lista, cual crees que es el caso principal?

Añadir el evento evidentemente cambia el estado por completo de nuestra aplicación.

Así que por supuesto es nuestro inicio obligatorio en la siguiente columna no tenemos ningún caso de uso que cambien el estado de nuestra aplicación pues no son más que casos de reportería, en la tercera columna si tenemos algunos elementos que cambian el estado de la aplicación, como editar el evento removerlo añadirlo al calendario y seguir un coach, pero editar no es un caso primordial porque es muy similar a añadir un evento por lo que añadir o editar tienen el mismo impacto en nuestro dominio, añadir eventos al calendario sin embargo si requiere extender el modelo de domino, para cada usuario necesitamos llevar un histórico de los eventos a los que ha marcado que asistirá así que este es otro caso principal y lo mismo aplica a seguir un coach para cada usuario necesitamos llevar un histórico de cada coach a los que el sigue.

En la cuarta columna tampoco tenemos casos principales pues todos no son más que reportes remover eventos del calendario y dejar de seguir un coach es similar a seguirlo y añadirlo al calendario por lo tanto no son más que extensiones. Así que tenemos que los más importantes son los siguientes:

Ahora bien, ¿qué pasa acá?, que el momento de escoger estos como casos principales, si deseamos mostrar nuestro avance al cliente, no estaríamos mostrando nada significativo, pues no hemos escogido ningún modo de reportería y no creo que el cliente encuentre chulo que solo se le muestre la base de datos por debajo.

En mi caso escogeré todos los próximos eventos, como actividad critica adicional, porque a partir de esta podemos ir mostrando un avance significativo del aplicativo, Pero, de igual forma en la cuarta columna voy a escoger uno de los casos mostrados ahí a los fines de que el cliente se vaya enamorando de la aplicación, pues aunque nosotros tengamos un enorme trabajo por detrás, si el usuario no ve por delante algo que valga la pena, se va a decepcionar y quizá no quiera pagarnos el adelanto o peor aún decida cancelar el proyecto.

Aunque no son casos verdaderamente principales, son casos de soporte, por eso me veo obligado a escogerlos. Recuerda siempre planear la pieza mínima de funcionalidad.

Posteriormente vamos a pasar mis casos principales a working, aclarar que en el mundo real hay metodologías más efectivas de trabajar este tema de los requerimientos, metodologías agiles, que no tienen nada que ver con velocidad, y normalmente se trabajan en entregables de dos semanas máximo, pero acá solo conceptualizamos esto como una forma organizada de trabajar.

## Iniciando las iteraciones

Ok, ya tenemos nuestros casos principales o el flujo lógico de desarrollo de nuestra aplicación, Ahora nos toca ponerlos en nuestro planeador para mantener un histórico de nuestro trabajo e ir trabajando de manera ordenada, lo cual vende mucho cuando estamos tratando de mostrar al mundo que sabemos o que somos duros.

Puedes usar cualquier herramienta como Microsoft Project que te permita llevar un histórico de los trabajos que vayas realizando en que estas trabajando actualmente y que aún falta por hacer, pero en el curso usaremos esta metodología.

Ahora nos toca esquematizar de manera básica como será la experiencia de usuario para nuestra aplicación a veces estas trabajando en un equipo donde puedes tener una persona dedicada al diseño pero la mayor parte del tiempo te toca trabajarlo a ti, lo que me interesa que se te quede de esta lección es que aprendas a tener el habito de tomar un pedazo de papel y dibujes como debe quedar tu flujo de trabajo dentro del aplicativo cuando este esté funcionando, que tu dibujo diga como el usuario va a navegar de una página a otra y como va a interactuar con los elementos que tiene en la página, como va a lucir la barra de navegación, etc. Es mucho más fácil dibujar esto en un pedazo de papel que ir directo al html y al css a tirar líneas que vamos a desbaratar una y otra vez, con eso en mente vamos a iniciar. No tires código sin un sketch

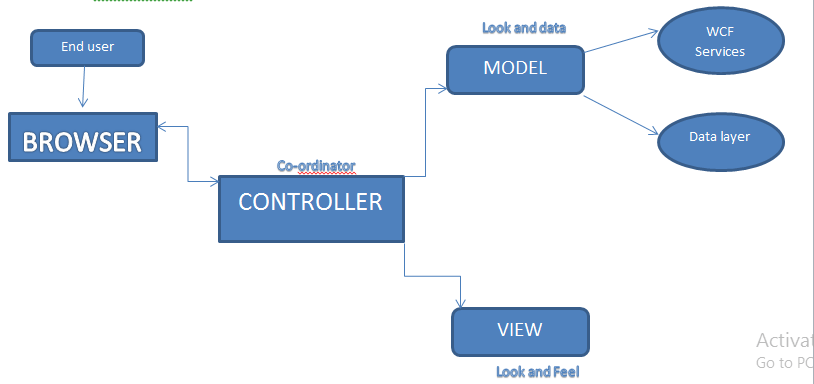
Debemos determinar cómo los usuarios van a navegar por la aplicación y recuerda que esta experiencia de usuario no tiene que ser perfecta solo enfócate en la interacción así que no intentes desglosar como va a lucir toda la aplicación completa con todas sus características, sino que ve construyendo sobre la marcha así que mantenlo simple a medida que vayamos evolucionando la aplicación la iremos haciendo mejor, así que lo primero que necesitamos es darle al usuario la habilidad de poder crear un evento necesitamos por lo tanto un formulario donde pueda añadir los detales del evento la pregunta es, ¿que debe pasar cuando el usuario haga clic en el botón? lo lógico sería redireccionarlo a la lista de los eventos para que pueda editar o eliminar si cometió un error pero no tenemos ese caso en esta iteración, nosotros tenemos ver todos los eventos, no solo los míos así que temporalmente lo vamos a redirigir al listado de todos los eventos que por ahora estará en la página principal

En esta lista al frente de cada evento pondremos un botón que dirá “deseo ir” con un signo de interrogación cuando el usuario haca clic en este botón el color y el texto deben cambiar y este será añadir a la lista de eventos a los cuales yo deseo ir es similar a como seguimos personas en twitter ahora, una vez el usuario lo haya añadido a su calendario debe tener la opción de verlo así que podemos dedicar una página y poner un link en la barra de navegación, de forma similar al frente de cada coach que está en cada evento tendremos un link o botón llamado seguir cuando hagamos clic en ese botón, cambiara de color y de nombre y el artista será añadido a la lista de quienes yo sigo añadiremos otro link en la barra de navegación llamado a quien sigo que tomara al usuario a la lista del coach que él está siguiendo.

Nuestra barra de navegación va a lucir algo como esto, tendremos el nombre de nuestra app que nos llevara al inicio y un link de logan que será reemplazado por el nombre una vez nos hayamos autenticado. Cuando haga clic en ese link debe aparecer un menú que solo estará disponible si el usuario esta logueado

Acá tendremos dos links el de ver los eventos a los que marque que planeo asistir y el de las personas a las que sigo, así como también un link adicional para salir del sistema o cerrar sesión.

# MVC

Para trabajar nuestra solución nosotros vamos a utilizar el patrón MVC, que no es más que un acrónimo que significa Modelo-Vista-Controlador, este no fue creado por Micrososft, es un patrón que tiene un tiempo cómodo incluso antes de que existiera la web, sin embargo es una metodología bastante practica a la hora de desarrollar y por eso ha tomado bastante auge, debido a la adopción por parte de diversos frameworks de este modelo como base.

Es bien sabido que en programación existe una cosa que es lo que el usuario ve, lo que esta en la pantalla, donde el usuario escribe o hace clics, esto no es mas que la vista, la vista es lo que se muestra al usuario mediante el navegador, luego tenemos el controlador, que podemos verlo como una especie de code behind con ciertas diferencias que veremos mas adelante, el controlador es el que se encarga de manipular los datos ya sea suministrándolo o ejecutando las acciones que el usuario haya enviado desde la vista, el controlador es el enlace entre el modelo y la vista, es el que toma el control cada vez que se marca un verbo en el navegador sobre una vista. Lo de los verbos lo veremos más tarde.

El modelo es la representación abstracta de las entidades de datos, es lo que usamos de base para trabajar y manipular datos. La idea del modelo, vista, controlador es separar y segregar la responsabilidad de cada cosa para evitar el código espagueti, que no es más que ese código que esta todo emburujado y que no hay forma de entrarle ni por delante con vaselina.

Un repaso breve de programación orientada a objetos

Un objeto