Extrayendo los principales casos de uso del requerimiento y creando las tareas.

¿Por dónde comienzo?

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 por lo 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,** además de que es un símil con la metodología ágil y es como se trabaja en la actualidad.

Aun cuando siempre estemos desarrollando para nosotros mismos, porque no tengamos clientes para vender la solución que deseamos construir, siempre, si queremos ser buenos desarrolladores, debemos meternos en el papel de cliente, para poder ver las cosas más allá del teclado e ir adquiriendo poco a poco la mentalidad de un ingeniero, recuerden que debemos programar “pa’ tías” y eso no se logra pensando como programador.

¿Hacer o no hacer prototipos?

Una práctica un tanto común en desarrolladores independientes, está la de hacer prototipos al cliente, donde se puede ver conceptualmente el sistema funcionando, estos se pueden hacer en Sketch, Paint, hasta una hoja de papel o pizarra, donde se detalla el flujo de trabajo que va a seguir el

aplicativo, sus pantallas y mas o menos como van a interactuar estos elementos.

Normalmente esto es bueno hacerlo si ya tienes el contrato amarrado con el cliente, nunca lo hagas si el trabajo no es tuyo, por más que el insista tu cliente, porque vas a invertir unas horas en hacer eso (mas el cumulo de horas que te tomó aprender a hacerlo, que es algo que se le debe sumar a cada coste), y al final el cliente no le va a gustar tu precio y con eso que le mandaste se va a un hindú que se lo va a hacer mas barato. Puedes cobrarlo por adelantado si es muy insistente.

Tampoco hagas el sketch de todo el sistema, solo haz el de tu mínimo producto viable, que es el que te va a ayudar a entender mejor el proyecto en la primera etapa.

Lo que si le puedes dar es una propuesta, donde detalles de manera general los módulos que has determinado que va a tener el sistema (donde pones la información que hayas recabado del cliente).

En los documentos adjuntos, pondré varios modelos de estos documentos, usted puede escoger el que mas le guste, yo no tengo un truco de cual funciona mejor o cual no, solo se que hay que tratar de ser claros y las restricciones deben estar determinadas desde un principio si es un proyecto en cascada, que admitámoslo, **la mayoría de los proyectos que vas a conseguir en la calle como freelancer, van a ser en cascada, pero te toca a ti trabajarlos como ágil.**

Documento de requerimientos.

Típicamente el desarrollo de software inicia con un **documento de requerimientos, que no es mas que el documento elaborado por el cliente, donde se contempla lo que el quiere que tu le realices mediante una aplicación de código, es la visión que tiene el cliente con el sistema que quiere que tú desarrolles.** 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 sean entregado estos, tu labor como ingeniero de software es extraer cuales son los casos 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.

*«**Mersy Core**: Es un sistema de expedientes clínicos para que los médicos de una institución o de su consulta privada, **lleven el registro clínico de sus pacientes**, los médicos o su personal, pueden **registrar los pacientes y agendarles citas,** a quienes se les asigna un **email único e irrepetible por cada organización** mediante un hash, pues **los pacientes son propiedad de las organizaciones y no se pueden cruzar informaciones,** a estos se les asigna una clave, mediante la cual pueden acceder al portal y en este tienen la posibilidad de **ver sus citas** indicando si van a poder asistir a una **consulta**, **previamente agendada esta en su calendario**, debe enviarse **notificaciones al paciente o el doctor en caso de cambios,** de parte y parte, en la cita, ya sea que el doctor canceló, modificó alguna información o el paciente sea quien la haya cancelado.*

***La consulta puede ser facturada** como una **orden** o **factura,** donde **los servicios pueden ser multi-precios dependiendo de la ARS**, facturas que pueden ser **cobradas, condonadas, o financiadas**, financiamiento que puede ser **saldado o abonado** hasta ser saldado.*

Los médicos, pueden crear posts, públicos, reservados o privados, para facilitar el compartir conocimiento, a través de su perfil, pueden ser seguidos o seguir a otros, los médicos a su vez pueden mensajearse entre ellos.

En una siguiente fase, se desearía que, los pacientes, pueden crear una cuenta personal, donde pueden asociar las diversas cuentas que tienen dentro del sistema Mersy Core y de esa forma gestionar todo con una sola cuenta, por lo que, no se puede crear una estructura que cierre o imposibilite esta posibilidad a futuro»

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 y cuales son las restricciones técnicas, por ejemplo en el último párrafo podemos ver que debemos desarrollar de una forma en la que sea posible que una cuenta maneje múltiples cuentas. Por lo que debemos trabajar sobre una base en la que esto sea posible a futuro, aunque no sea un proyecto que nos vayan a asignar a nosotros.

(En Construcción) Levantar el documento de requerimientos

Casos de uso

Los casos de uso o historias se expresan con pocas palabras, pero claras y concisas, que no admitan ambigüedades o que se presten a interpretación. Dependiendo de la metodologías y nivel de organización, pueden definirte cuales son las entradas y salidas esperadas, y cuáles son los criterios de aceptación. En este caso no vamos a llegar a esos niveles, porque somos nosotros mismos quienes vamos a desarrollar y tenemos bastante experiencia en desarrollo (aunque no lo creas, si leíste las primeras páginas de este material, tienes más experiencia en desarrollo que gente con 3 y 4 años programando).

Aunque parezca irrisorio, Twitter ayudó mucho al desarrollo de software, porque enseñó a la gente a concretizar y a resumir; claro, también creó una cultura de inmediatez muy absurda, pero eso no son temas de esta clase.

En el primer párrafo podemos identificar dos casos de uso principales, así como quien no quiere la cosa, sin tener que analizar mucho: **“Crear un paciente”** y **“crear una cita”.** Si pensamos un poco más, podemos ver que tendríamos **“crear cuenta de usuario”** y **“Agendar cita”.**

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 que tu puedas sacar del requerimiento y en breve compararemos tu solución con la mía.

Vamos entonces a identificar nuestros casos.

Casos de Autenticación

- Módulo de registro (creación de cuenta).
- Módulo de acceso (Login).

- Cerrar sesión (Logout).
- Módulo de cambio de contraseña.
- Edición de perfil.
- Registrar usuarios.
- Modificar, eliminar usuarios.
- Configurar niveles de acceso y permisos.

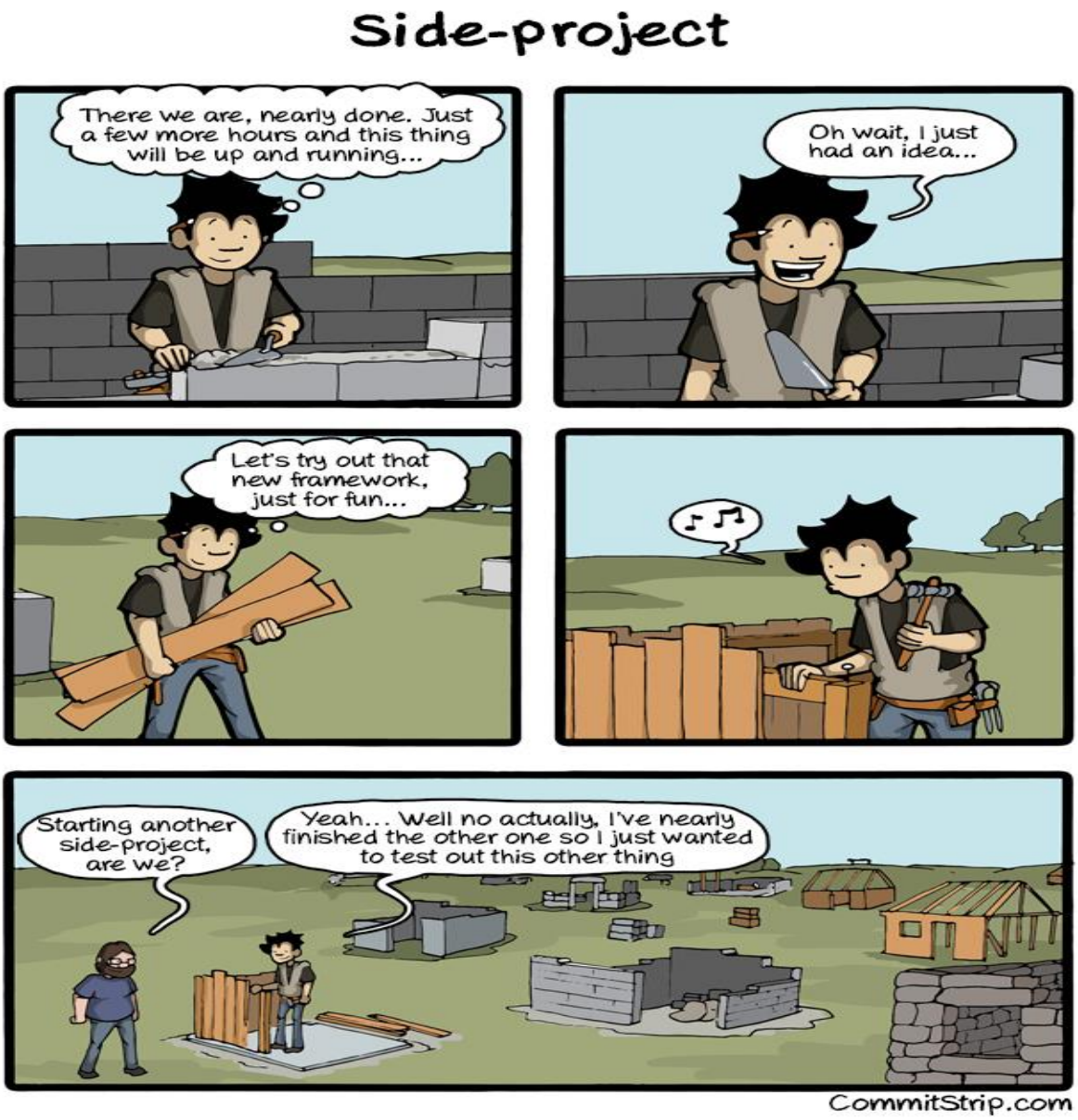
Acá podemos ver algo interesante, a pesar de que en el documento de requerimientos no se me dijo que debía crear una pantalla de acceso, es obvio que hace parte de nuestro proyecto, por lo que muchas veces como ingenieros, **corresponde a nosotros agregar esos casos o pasos, sin los cuales no podemos 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 o porque aprendimos con el profesor Germosen estos atajos que permiten ver más allá de lo que está escrito en el papel.

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 mientras vas construyendo tu sistema para llenar esos vacíos existenciales y dudas que tengas.

En este caso usé mi criterio para determinar que: si tengo un registro, por lógica, debo tener un login, log out, etc. Eso y que yo soy mi propio cliente para este proyecto. lol

Reusar o Reinventar



La buena noticia con respecto a la primera parte de los casos de uso es que net core, mediante la modularización, tiene formas muy sencillas de construir un sistema de autenticación y registro, basado en Identity, que tu puedes extender para agregar aquello que vayas necesitando, según lo requiera la trama.

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, reutiliza siempre que se pueda, no solo con la autenticación, sino con la mayoría de componentes que tú te veas en la necesidad de crear, busca si no hay algo ya creado, sácate de la cabeza eso de que tú tienes una super mega lógica que nadie ha imaginado para controlar los usuarios o llevar un control de eso que quieres hacer, que solo tú en el mundo sabe cómo hacerlo bien, es probable que ya exista, busca y rehúsa y si no existe, una vez lo hagas, comparte.

Casos de Paciente

- Añadir un paciente.
- Listar mis pacientes.
- Editar y eliminar un paciente.
- Ver todos los pacientes del centro u organización.
- Buscar pacientes por distintos filtros.
- Ver los detalles de un paciente.

Casos de Cita

- Añadir citas a un paciente.
- Listar citas del paciente.
- Editar y eliminar una cita.
- Ver todas las citas del centro u organización.
- Buscar citas por distintos filtros.
- Ver los detalles de una cita.

Casos de Calendario

- Añadir citas a tu calendario.
- Añadir citas al calendario del paciente.
- Modificar y remover citas de tu calendario.
- Modificar y remover citas del calendario del paciente.

A pesar de que el tema de añadir los citas a un calendario no eran parte de los requerimientos originales, 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 picapollero, 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.

Siempre vamos a ver cosas que no estaban en el documento de requerimientos original, pero como somos super analistas podemos prever algunas de ellas y sugerirlas para que haya mas confianza en que se escogió a un excelente desarrollador en lugar de ponernos a refunfuñar y maldecir o decir que el cliente es un imbécil.

Ahora bien, **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.

Casos de Historia Clínica (Registro Clínico)

- Añadir información clínica a un paciente.
- Listar información clínica de un paciente.
- Editar y eliminar información clínica de un paciente.
- Ver los detalles de la historia clínica de un paciente.

Casos de Visita (Consulta)

- Añadir información de visita a la historia clínica de un paciente.
- Listar visitas de la historia clínica de un paciente.
- Editar y eliminar visitas de la historia clínica de un paciente.
- Ver los detalles de una visita de la historia clínica de un paciente.

Casos de Ordenes

- Añadir órdenes a un paciente.
- Listar ordenes de un paciente.
- Editar y eliminar ordenes de un paciente.
- Buscar ordenes por distintos filtros.
- Ver los detalles de la orden de un paciente.
- Combinar diversas ordenes y hacerlas una sola factura.

Casos de Facturación

- Añadir facturas a un paciente.
- Listar facturas de un paciente.
- Editar y eliminar facturas de un paciente.
- Buscar facturas por distintos filtros.
- Ver los detalles de la factura de un paciente.

Casos de Servicios

- Añadir servicios.
- Listar servicios.
- Editar y eliminar servicios.
- Ver los detalles de un servicio.

Casos de Precios

- Añadir precio a un servicio.
- Listar los precios de un servicio.
- Editar y eliminar precios de un servicio.

Casos de Acuerdos de Pago (Préstamo o financiamiento)

- Añadir acuerdo de pago a factura.
- Listar información de acuerdos de pagos de un paciente.
- Editar y eliminar de acuerdos de pagos.
- Ver los detalles de un acuerdo de pago.

Casos de Pagos

- Añadir pagos a una factura.
- Añadir pagos a un acuerdo de pago.
- Listar pagos realizados por un paciente.
- Eliminar pagos de una factura.
- Ver los detalles de un pago.

Casos de Posts

- Añadir posts.
- Listar posts.
- Buscar posts por diversos filtros y criterios.
- Editar y eliminar posts.
- Aprobar posts públicos.

Casos de seguimiento

- Seguir un doctor.
- Dejar de seguir un doctor.
- Ver a quienes sigo.
- Pagina de inicio basada en los posts de aquellos a los que sigo.
- Ver los detalles de un pago.

Casos de Mensajería

- Enviar mensaje a doctor.
- Eliminar mensaje.
- Listar chats

Dependencias de casos y Ruta critica

Una de las mejores enseñanzas que se pueden aprender solo en la Universidad, porque en internet nadie la considera importante, es en investigación de operaciones, la famosa **ruta crítica**, **que no es más que determinar el tiempo y presupuesto de ejecución en nuestro proyecto, cual es el camino por donde debe seguir la construcción, según las**

dependencias que se vayan suscitando, o el orden de prioridades, claro, nosotros solo vamos a usar una parte de lo aprendido en investigación de Operaciones, pues, mientras en esta el trabajo se enfoca en que se puedan ir haciendo actividades en paralelo que no tengan dependencia (como ir mezclando los huevos en lo que se cocina el arroz), nosotros como desarrolladores, nos vamos a enfocar únicamente en la ruta crítica para entregar un producto mínimo viable en la brevedad de tiempo posible y luego vamos anulando procesos, según lo vayamos coordinando con el cliente.

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

Por poner un caso, ¿podemos cancelar una cita si primero no hemos creado una?, ¡Claro que no!, ¿Cierto?

Lo que quiere decir que, por ejemplo, tenemos que implementar primero la función de crear pacientes para luego crear citas que posteriormente puedan ser canceladas.

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

Te muestro mi solución en un documento de Excel que está en el repositorio llamado Flow_work.

Si tu solución se parece a la mía, es genial, pero si no, 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.

Yo lo organizo de esta forma, por lo regular en una pizarra, porque facilita mi comprensión, tu puedes perfectamente ponerlo en Project, y hacer algo más profesional si así lo deseas.

Ahora es momento de identificar nuestra **ruta crítica estos casos de uso principales y son los que le dan forma al dominio de nuestra modelo de datos este es el que involucra la captura de data y cambia el estado de la aplicación.**

Así que, de esta lista, ¿cuál crees que es el caso principal?

Añadir pacientes 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 paciente, removerlo y agendar una cita, pero editar no es un caso primordial porque es muy similar a añadir, por lo que añadir o editar tienen el mismo impacto en nuestro dominio, añadir cita al calendario sin embargo si requiere extender el modelo de domino, para cada usuario necesitamos llevar un histórico de las citas que ha creado así que este es otro caso principal.

En la cuarta columna tampoco tenemos casos principales pues todos no son más que reportes.

Remover citas del calendario es similar a crearlo por lo tanto no son más que extensiones.

Así que tenemos que los más importantes son los indicados en el documento “xApplicationFlowx”

Si no entendiste muy bien esta parte, es porque eres muy bueno a nivel técnico, tranquilo, no pasa nada, imagina que la ruta critica es la que te va a ayudar a determinar cuáles son las tablas (dominios) que vas a necesitar y según vayas viendo que necesitas tablas para realizar

dichas interacciones, esa es tu ruta crítica, fin, quédate con esa información.

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 un select a la base de datos y le digamos: “mire ahí sus datos, mire que todo funciona y se guarda bien”

En mi caso escogeré listado de pacientes, 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.

Si nosotros no nos sentáramos a realizar este esquema de trabajo, iniciaríamos tirando un dominio completo, con todas sus funcionalidades esperadas, y pasan las dos semanas y ahí lo único que le mostramos al cliente es que logramos hacer un CRUD de pacientes y cuando el quiere ver más, ah no, no haz hecho nada porque tenias tiempo de hacer 4 pantallas y esas que pudieron ser crear paciente, ver pacientes y agendar citas a paciente, se quedaron solo en un CRUD que no dice nada.

Por esto, nunca inicies por el código, siempre por el papel.

El resultado final lo puedes ver en el documento “ApplicationFlow”

Tareas

Basados en los casos de uso es que se hacen las tareas en esta primera etapa de desarrollo, ya a medida que vayamos metiendo mano, las tareas

van a estar más refinadas, por lo pronto, lo único que hago, al menos en mi caso, es copiar todos estos casos y remover la palabra caso, dependiendo de ciertas circunstancias podemos agregar algo de información extra, y luego las pego en una la plantilla de visión de proyecto.

Un documento de visión del proyecto es el norte de este, la idea que se tiene para con el proyecto, es el timón que va guiando hacia buen puerto la embarcación, en este se van asignando puntos de criticidad, fechas de compromiso, comentarios y estatus.

Pero esto no es nada recomendable, pues no te puede dar estadísticas del tiempo que te toma cada tarea y esto no es medible y recuerda que lo que no se mide, no se mejora, pero, hay caprichos de viejito que son difíciles de dejar.

En este documento también se van subiendo las restricciones que se van dando o aclaraciones que se van realizando en diversos temas, según vaya teniendo conversaciones con el cliente. La configuración escogida para las tablas a modo de diccionario de datos, las fechas de compromiso con el cliente, etc.

Listar casos como tareas en algún planner.

Ya con nuestros casos identificados, podemos usar alguna herramienta de planificación de trabajo, puede usar Assana u otro, también puedes usar el mismo GitHub ya sea con tareas o issues, algún bloc de notas como están la mayoría de mis proyectos por caprichos de viejito o con lo que te sientas cómodo, yo en lo personal,

Pueden ver el documento en el repositorio, el cual en su primera versión tiene la explicación de cada renglón, el resto de las versiones solo lo tendrá como comentarios de tooltip.

Priorizar las tareas

Posteriormente vamos a marcar la priorización de mis casos, donde cinco asteriscos van a ser aquellos casos prioritarios según la ruta crítica, tres los que yo considere que son de vital importancia, dos los que no son tan importante, pero deben ser trabajados pronto y uno o vacío, los que se

pueden dejar para el final, ya sea porque nadie depende de ellos o no son importantes por el momento, mientras que 4 son los casos que el cliente indica que son su prioridad.

He de 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 antes de determinar que se va a trabajar, se pesan en nivel de dificultad cada tarea y de esta forma se determina, según la capacidad del equipo, cuantas tareas pueden entrar, sin embargo acá solo conceptualizamos esto como una forma organizada de trabajar.

Crear tareas a trabajar en el backlog de GitHub

Una vez determinado cuales son las tareas prioritarias, debemos crear estas tareas en GitHub, asignadas al proyecto (o sprint) en el que nos encontremos.

Esquematizando las iteraciones

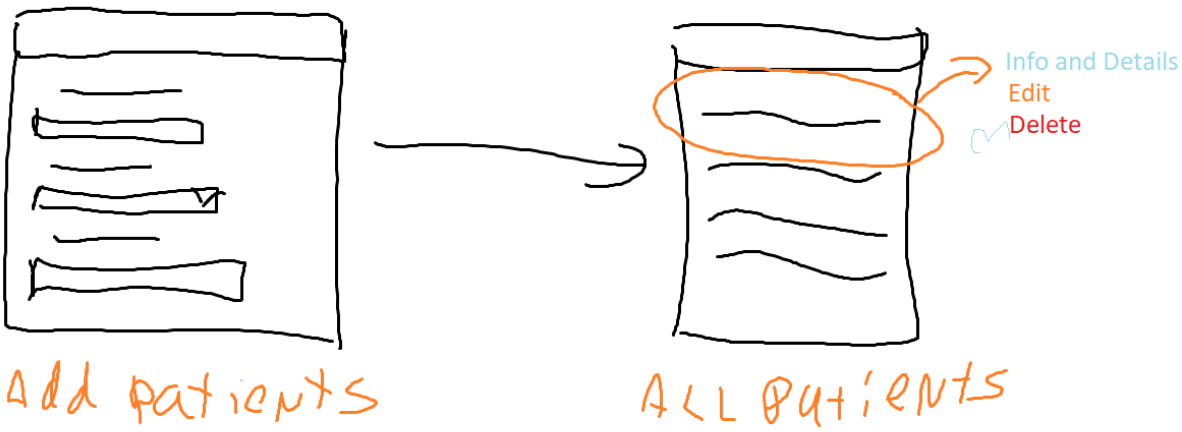
Ahora nos toca esquematizar de manera básica cómo será la experiencia de usuario para nuestra aplicación, a veces estas trabajando en un equipo donde puedes tener una persona dedicada al diseño pero la mayor parte del tiempo te toca trabajarlo a ti, lo que me interesa que se te quede de esta lección es que aprendas a tener el hábito de tomar un pedazo de papel o una pizarra 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.

Se que puedes considerar que esto es una perdida de tiempo, porque a medidas que vayas desarrollando, tu esquema inicial va a cambiar mucho, sin embargo, te aseguro, que, si no se queda al menos un 60% del esquema original que hiciste, es porque lo hiciste mal o te falta experiencia.

Cuando tu ideas un esquema, siempre vas a poder darte cuenta desde un principio, de algunos errores que pudieras cometer si saltas directo al código, no lo veas como perdida de tiempo, que no lo es, te lo garantizo.

así que lo primero que necesitamos es darle al usuario la habilidad de poder crear un paciente, necesitamos por lo tanto un formulario donde pueda añadir los detalles del paciente la pregunta es, ¿que debe pasar cuando el usuario haga clic en el botón guardar? Pues que nos redireccione al listado de mis pacientes, que, si bien se supone que debe filtrar solo los pacientes de mi organización, inicialmente vamos a dejar esta funcionalidad para más tarde.



En esta lista al frente de cada paciente pondremos un botón que dirá “info y detalle”, el cual nos llevará al detalle del paciente, donde podremos agendar citas, editar y eliminar, pues en lo personal, no me gusta poner estas opciones en el listado.

No importa que tan feo sean, siempre que tu y tu cliente, las puedan entender. Las demás no las haré por un tema de espacio, pero el punto es que ya entendiste la idea.

Nuestra barra de navegación va a lucir algo como esto, tendremos el nombre de nuestra app que nos llevará al inicio y un link de login 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 este logueado.

Acá tendremos diversos links con varias funciones, donde una de ellas es la de cerrar sesión.

Resumen

- En este módulo vimos:
- Por donde iniciar
- Como recabar información para tener un documento de requerimientos de calidad.
- Trabajar con el documento de requerimientos.
- Identificar los casos de uso y la ruta critica
- Esquematizar las interacciones.

Asignación 3

1. Crear el documento de requerimientos del proyecto que deseas trabajar en paralelo con el nuestro, recuerda que la mejor forma de

aprender no es directamente imitando lo que hace el profesor, sino ir experimentando con tu propio sistema.

2. Extraer los casos de uso y los casos de uso principales (ruta critica).
3. De estos casos de uso extraer las tareas e insertarlas en algún planeador o plantilla de visión de tu proyecto.
4. Copiar en Github, las tareas de la ruta critica que nosotros deseamos trabajar en esta primera iteración.