CAPITULO 2: VACIANDO LA TAZA

cuanta más experiencia tengas, más te costará deshacerte de tus malos habitos
lo pero de todo es no darse cuenta de la ignorancia de uno

1. tu primer lenguaje
    - usa ejemplos de tu trabajo para encontrar un nuevo trabajo
    - usa test para aprender un lenguaje
    - usa test para aprender una api
    - la mejor forma de aprender un lenguaje, es tener cerca a un experto en él
    - no dejes que tu experiencia con un lenguaje te ate para aprender nuevos
2. cinturon blanco
    - debes desaprender lo que has aprendido, debes estar abierto a nuevos conocimientos (white belt)
    - la libertad de parecer loco/tonto/excentrico es una de las claves del exito de los genios
    - debemos dejar de lado conocimientos y experiencias previas para ahcer sitio a nuevas
    - Jerry Weinberg: si no dejas de lado lo que haces bien, nunca harás grandes progresos
3. da rienda suelta a tu entusiasmo (unleash your enthusiasm)
    - curiosidad y 'excitement' insaciables
    - no dejes que nadie te apague tu entusiasmo por la artesanía (del soft)
    - si tu equipo no alimenta tu entusiasmo, debes encontrar otros medios de alimentarlo (nurture your passion)
    - cuestiona todo, el porqué de todo
4. habilidades concretas
    - aprende a hacer algo bien y que sea útil para el equipo, así te aceptarán mejor
    - estas habilidades debes poder mostrarlas en una entrevista, como jugutes, y deben responder a la pregunta: "si te contrato hoy, qué puedes aportar mañana que nos beneficie"
5. expón tu ignorancia
    - déjales verte crecer.
    - los artesanos construllen su reputacion mediante fuertes relaciones con sus clientes y colegas
    - di la verdad
    - tu reputación se construirá según tu capacidad de aprender, no sobre loq ue ya sepas
    - haz preguntas
    - mostrando tu ignorancia, muestras tu capacidad de aprender
    - este proceso de aprendizaje es ARTESANÍA
6. enfrentate a tu ignorancia
    - haz lo que sea para aprender las cosas que no sepas: articulos, faqs, blogs, juguetes, almas gemelas, mentores, ...
    - debes trabajar por el bien del equipo y comunidad, no por el propio interes
7. salta al vacío, el gran salto, el lejano final ¿? (deep end)
    - cuando te ofrezcan un rol de perfil alto o un problema difícil, agárralo con las 2 manos
    - estate preparado para fallar, y para ver nuevas oportunidades en ese fallo, oportunidades que lo timidos nunca verán
8. retrocede hacia las cosas que dominas
    - si estás abrumado por 'expon tu ignorancia', puede que sea hora de hacer algo con lo que te encuentres cómodo, analízalo y dáte cuenta de hasta dónde has llegado
    - retrocede hasta donde dominas para tomar impulso

- a veces, necisitas volver un paso atrás para dar una gran zancada
- no abuses de esto o tu experiencia estará obsoleta (todas las actuales lo estarán tarde o temprano) y tendrás que volver a empezar

CAPITULO 3: ANDANDO EL LARGO CAMINO

1. el largo camino
   - la última moda repite los mismos errores que se cometieron hace mcho tiempo
   - valora el aprendizaje y oportunidades de crecer a largo plazo sobre tu salario o sobre liderazgo de la forma tradicional
   - nada está cerrado a ti: ni negocios ni tecnologías
   - la longitud del viaje, multiplica las posibilidades de encontrar nuevos caminos
   - el camino te enseñará la verdad, y te ayudará a transcender la tecnología, llegando al fondo de los problemas
2. artesanía mejor que arte/belleza
   - enfócate en añadir valor a tu cliente sobre avanzar sobre tus propios intereses
   - desarrolla la habilidad de añadir utilidad antes que belleza
   - trabajar en problemas reales para gente real es lo que hace honra la artesanía
3. motivaciones sostenibles
   - la mejor motivación es programar por programar
   - si quieres ganar dinero, te tendras que enfrentar a problemas que la gente esté dispuesto a paga por ellos
   - problemas tediosos, definidos vagamente, innecesariamente complejos, difíciles o desconocidos. burocracia, dificultados personales y un terrible liderazgo. son obtaculos que te encontraras
   - motivaciones: dinero, programar por programar, construirte una reputacion
   - haz lo que más te guste, el dinero vendrá despues
4. alimenta tu pasion
   - trabaja en lo que te gusta
   - busca almas gemelas
   - estudia a los clásicos
   - dibuja tu propio mapa
   - define claramente el tipo de entorno que quieres trabajar: sal de una reunion que no te interesa, no liberes código que huela mal, no hagas horas extras, ...
5. dibuja tu propio mapa
   - identifica un objetivo lógico, pero ambicioso, en tu carrera
   - llegar ahí, depende solo de tí
   - debes re-configurar tu mapa según las condiciones van cambiando
6. usa tu titulo
   - usa tu titulo para evaluar tu empresa, no a tí mismo
   - no creas en un titulo rimbombante
7. permanece en las trincheras
   - antes de aceptar promociones que te alejen de programar busca otras recompensas: dinero a formas no tradicionales de liderazgo: asesoría, ...
8. un camino diferente
   - es problable que haya gente que se salga del camino, pero si algún dia vuelven, recibelos con los brazos abiertos, sus experiencias fuera del mundo del software seguro serán muy valiosas
   - no tengas miedo de hacer cosas diferetnes en tu vida

CAPITULO 4: AUTORECONOCIMIENTO

es facil superar a los mediocres en el mundo del desarrollo del software porque demasiada gente se contenta con estar por encima de la media

tu meta debe ser medir tus habilidades para encontrar caminos de ser mejor que ayer

la humildad es una de las bases del aprendiz, combínala con la ambicion

1. sé el peor
    - rodéate de desarrolladores mejores que tú
    - pertenecer a un equipo fuerte the hará sentir que trabajas mejor
    - como eres el miembro más débil, deberás ser el que más se esfuerze
    - ten cuidado, no caigas en la trampa de sentirte mal porque eres el más débil
2. encuentra mentores
    - lo primero es encontrar un maestro del cual aprender
    - todos estamos andando el largo camino, luego, nadie sabe todo sobre el desarrollo del software
    - no te sientas intimidado por preguntar a un desconocido si quiere ser tu mentor
    - así como tú buscas mentores, llegará un momento en el que otros quieran que tú seas su mentor, no lo rechaces.
3. almas gemelas
    - debes buscar gente como tú, que busca la excelencia
    - durante los años de aprendizaje, necesitarás compañeros que caminen junto a tí, no solo maestros a los que seguir
    - complementa tus mentores con tus almas gemelas (tu comunidad)
    - fuérzate a hacer preguntas que hagan pensar a tu comunidad
4. trabajar codo con codo
    - crea ocasiones para trabajar con un desarrollador en la misma tarea y hazlo junto a él/ella
    - siempre habrá pequeños trucos que sólo podrás aprender trabajando junto a un compañero
    - si no puedes trabajar directamente con ellos, puedes "mirar" cómo trabajan
5. limpiar el suelo
    - al comenzar en un proyecto, ofrécete voluntario para tareas aburridas, simples y no glamurosas, pero necesarias de todas formas
    - ten cuidado, no caigas en la rutina de solo hacer ese tipo de tareas, este patrón te debe servir para mostrar tu aptitud para aportar valor al proyecto

CAPITULO 5: APRENDIZAJE PERPÉTUO

el desarrollo de software se divide en dos actividades principales: aprender y comunicación

el aprendiz debe 'aprender a aprender', es decir, a ser humilde y a dejar de lado cierta experiencia para poder aprender cosas nuevas y rompedoras

1. expande tu radio de actuación
    - debes desarrollar la habilidad de absorver nueva información, así como entenderla, retenerla y aplicarla
    - qué te ayudará: suscríbete a blogs de desarrolladores, síguelos en twitter, subscríbete a listas de correos y trata de responder las preguntas de los demás reproduciendo sus quejas, únete a un grupo local, convence a tu jefe de que te mande a conferencias, se agradecido con los autores de los libros que lees
    - este patrón acelerará tu aprendizaje, pero no lo uses durante mucho tiempo
2. practica, practica, practica
    - tómate tu tiempo para practicar en un entorno en el que te sientas cómodo si fallas
    - la forma ideal es que un mentor te asigna ciertas tareas acordes con tu nivel, valore tu eficacia y decida contigo los siguientes ejercicios
    - "en el desarrollo del software, practicamos en el trabajo, es por eso por lo que cometemos los fallos en el trabajo"
    - necesitas retroalimentación cuanto más temprana mejor, si no, es posible que desarrolles malos hábitos

3. juguetes rompibles
- desarrolla pequeños sistemas similares a tu trabajo (pero no tan complejos) y permítete fallar en el diseño o la implementación
- estos juguetes son reimplementaciones de estándares: wikis, blogs software, juegos (tetris, ...)

4. usa el código fuente
- busca código de otra gente y léelo, léelo como si fuera un libro
- busca gente a tu alrededor que esté interesada en leer tu código, si consigues apreciar su opinión, serás un mejor programador
- la mejor manera de aprender es leer código de proyectos open source

5. muestra cómo trabajas
- sé crítico contigo mismo, si algo no te gusta, pregúntate el por qué es así y si hay alguna forma de solucionarlo
- crea un Mapa de Prácticas Personales, eso te mostrará claramente cómo trabajas
- tu meta debe ser llegar a ser habilidoso, no experimentado. Experimentado quiere decir que llevas mucho tiempo en lo mismo

6. registra lo que aprendas
- mantén un registro de lo que aprendas, en una wiki, un fichero de logs, blogs, lo que sea
- no caigas en la trampa de dejarlo escrito y olvidarlo

7. comparte lo que aprendes
- si cuando estas aprendiendo algo, no hay un manual, créalo tú

8. crea lazos de retroalimentación (feedback loops)
- aquellos que no tienen habilidades, no suelen ser conscientes de ello
- solicitando información pronto, frecuentemente y eficazmente te mantendrás informado sobre si vas por el buen camino o al menos sabrás lo incompetente que eres y podrás corregirlo
- mecanismos: test driven development, revisiones de código, programación en pareja, pregunta a la gente qué piensan de tu trabajo

9. aprende a diferenciar cómo fallas
- la meta es conseguir discernir la forma, condiciones, hábitos y comportamiento que te conduce a fallar
- no puedes ser perfecto, así que valora qué cosas quieres mejorar y qué cosas vas a tener que dejar de lado. para ello te puede venir muy bien dibujar tu propio mapa

CAPITULO 6: CONSTRUYE TU CURRÍCULUM

1. Lista de lectura
- deberías mantener una lista de lectura, con los libros que quieres leer y los que has leído, para no olvidarlos

2. lee constantemente
- enfoca tu sed de aprender en consumir todos los recursos escritos que puedas, dando preferencia a los libros frente a los blogs

3. estudia a los clásicos
- usa libros "antiguos" para aprender y los blogs y otras experiencias para ver cómo han evolucionado esos conceptos

4. profundiza
- aprende a profundizar en las herramientas, tecnologías y técnicas, adquiere el conocimiento necesario para poder explicar el por qué las cosas son como son
- profundizar en una tecnología te permite explicar qué es lo que hay debajo de la superficie, esto te diferenciará de muchas personas con las que trabajas
- ser capaz de leer especificaciones así como código quiere decir que no hay nada que se te pueda escapar
- toma tu información de fuentes originales, no del blog del blog del comentario del libro no se cual, si no la especificación de la tecnología

- cuando leas un tutorial, no busques código para copiar, busca una estructura que encaje en tu nuevo conocimiento

5. herramientas familiares

- a veces, la mejor herramienta no es la que tú conoces. debes ser flexible y usar las herramientas que usa tu equipo
- ten cuidado y no caigas en la tentación de pensar que tus herramientas pueden solucionar todos los problemas (golden hammers)
- dejar ir herramientas familiares es duro, pero es algo que debes aprender

CAPITULO 7: CONCLUSIÓN

la mayoría de los programadores piensan que están por encima de la media, pero la verdad es que la mayoría está por debajo