-progressive overload
-imagine o uso do app como um ping pong entre "*o que preciso vs o que posso fazer*". o primeiro o usuário já sabe, e é afetado pelas possibilidades que a tela exibe. poluição visual não se restringe a outdoors.
-encontrar alguém na multidão é difícil, e é exatamente o que ocorre quando o app está cheio de features. na verdade, do mesmo jeito que "não existe LP longa, apenas LP chata ou sem sentido/não-linear".
-o mesmo vale aqui. não existe tela lotada, existe tela confusa, se os itens não forem claros o suficiente podem confundir uns aos outros, e é por isso que existe a Lei de Hick. os itens devem ser claros e estarem relacionados. Devemos fazer correlações entre elementos, agrupá-los e correlacionar os grupos.
- Ex: no Tinder temos like e dislike, junto com a foto, no objetivo de exibir os perfis.
- Ex: no Uber temos um input e um mapa, no objetivo de definir o destino do usuário. nessa ideia, seria interessante também um speech-to-text, mostrar destinos recentes ou recorrentes, sugerir pegar o texto da área de transferência, etc.
-trabalhamos com imagens e conceitos. marcas passam ideias, defendem conceitos, símbolos, representam algo. tem a ideia de causa, inimigo comum, solução de problemas ou algo mais "sexy canvas" como segurança, vaidade, etc. aqui não é diferente, o usuário se situa na tela montando um [modelo mental] que ele usa para se posicionar, processar e tomar uma ação.
-lembre-se que estamos no ping-pong de "quer vs pode" fazer, então, usar um modelo mental pronto (principalmente se for de algo já do nicho), agiliza o processo e diminui o "overhead" ou "overload" do cliente, já posiciona ele em um nível de consciência alto e ele se sente mais a vontade usando o app, tem uma experiência mais agradável, menos medo de errar, se situa melhor, etc.
-o progressive disclosure atua como uma solução natural para esse problema, pois permite usuários (iniciantes ou avançados) focarem nos core features da aplicação e deixarem as features extras em uma tela separada. dessa maneira não só agrupa as features em labels/modelos mentais diferentes de "primárias/secundárias", como elas irão se complementar entre si e um grupo não interfere no outro, tornando mais simples o uso, facilitando o ping/pong e diminuindo a chance de erro do usuário.
-pense como um HTTP request. se o HTTP tiver um request body muito longo, provavelmente contém vários dados que englobam "valores" diferentes envolvendo relacionamentos do banco e etc, é muito mais informação para processar e com isso o request demora mais. enquanto que ter requests "focados", facilitam o processamento, a implementação, e diminuem o tempo de resposta.
-o mesmo acontece no ping-pong. o usuário bate o olho na tela, tem contato com várias informações para processar e isso complica para ele se situar. o SIGAA é horrível por conta disso.
-um modelo mental bem comum é o de toolbars usando o "right-click" do mouse, pode ser interessante implementar isso. principalmente no escudero. em vez de ter um modal ou select ocupando espaço na tela (que só seria usado 30% das vezes e polui para o usuário), ele pode ficar escondido pelo botão direito, aparecendo apenas "on-demand", seguindo o progressive disclosure.
-porém é uma faca de dois gumes, nem sempre o usuário iniciante vai deduzir que tal feature esteja "escondida". é bom deixar as features primárias expostas e/ou fazer um onboarding.
-a dopamina é o carro-chefe do nível de consciência. benefícios e dores evitadas "ativam" o usuário, despertam o interesse nele, faz ele se situar no nível de consciência e continuar explorando o app, de acordo com a necessidade, dores evitadas, etc.
-se o aplicativo for fácil e tiver linearidade/continuidade, o usuário vai explorando elas tranquilamente. se não, e gera confusão e frustração, afetando a experiência do usuário.
-por outro lado, um aplicativo necessário para a pessoa, como um aplicativo de consultas médicas e aplicativo de banco, podem ter uma experiência horrível que o usuário ainda vai usar, contra a vontade:
- ex: aplicativo caixa tem, do auxílio emergencial 2020.
-justamente por isso o progressive disclosure + mapas mentais são importantes, juntamente com o familiarity bias. se o usuário tiver que aprender, deve ser dopaminérgico, então temos duas dicas: usar um **rápido** tour ou *onboarding*, e/ou destacar os benefícios de passar pelo tutorial, usar gamificação, etc.

-na verdade o progressive disclosure (como no Tinder) já é um meio de fazer o usuário aprender enquanto usa, então já remove o overhead inicial.
-o user engagement level está diretamente ligado ao N.C.
-é possível hackear o user engagement com a estratégia do tinder, pagar para sair na frente se houver ROI. exemplo, o Photoshop tem 3 versões, uma para cada nível de consciência: CS, Elements, Starter. a mais complexa vem com todas as features (N.C alto), tem treinamentos e livros mais caros que a ferramenta, mas como é voltada para profissionais que vendem ensaios, cursos e etc, o ROI é alto, então elas pagam para aprender mais rápido e hackear o engagement. para uma pessoa que quer editar rapidamente uma foto para postar no Instagram, não faz sentido.
-existe um balanço entre ROI, dopamina e user engagement. se o site for facilmente substituível, fica mais tangível e benéfico (além de evitar dor) caçar outro site em vez de permanecer quebrando cabeça em um site. é o princípio por trás do DEW e apps de banco. daí também vem a frase: usuário não lê, escaneia.
-o tempo gasto em um site é em média 30s e no máximo o usuário gasta 2min indo fundo no site. é por isso que as headlines e primeira dobra devem ser fortes: clarifica para o usuário porque ele clicou e motiva ele a permanecer na página, aplicando curiosidade para o resto, aplicando progressive disclosure para o resto da página, e sendo único para evitar o *bounce*.
-