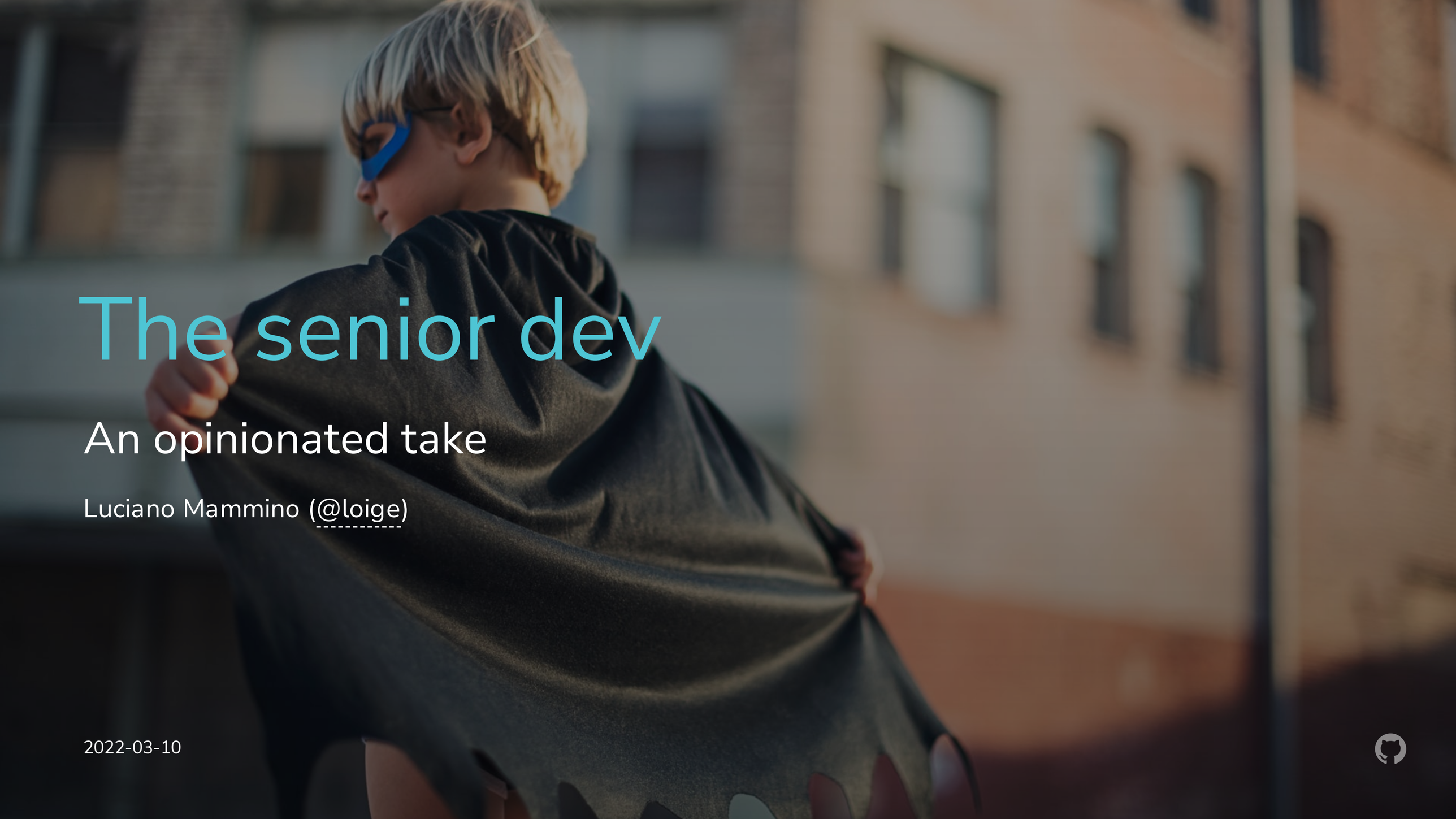


The senior dev

An opinionated take

Luciano Mammino (@loige)

2022-03-10

Agenda

- Discuss what is expected from a **senior software engineer** (skills, mindset, duties)
- Packed with *opinions** (my own and m o re)
- For devs **to be inspired**
- For managers to **hire**, **set expectations**, **support**, and **evaluate**

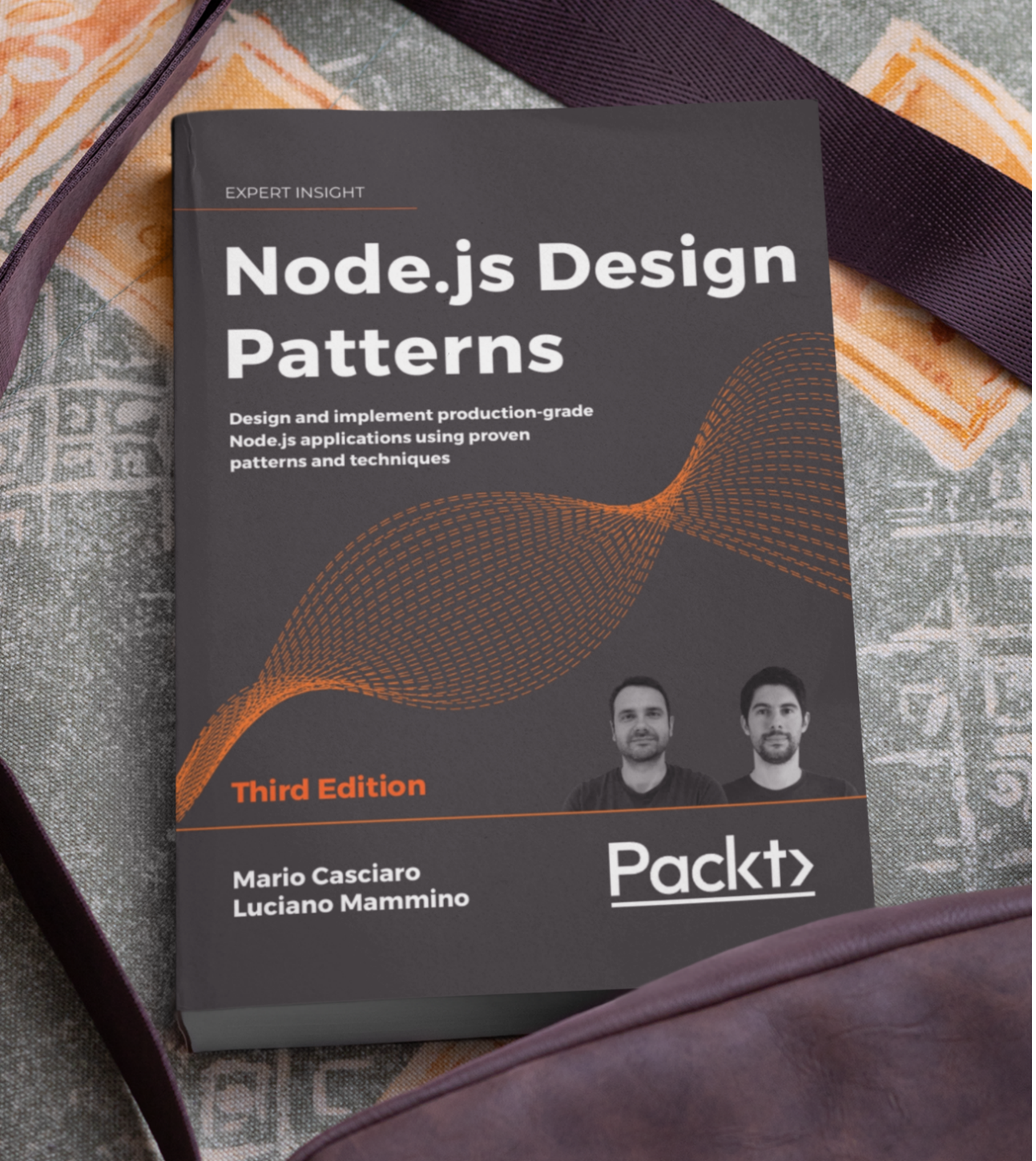
* opinions are subjective

Photo by Alexander Milo on Unsplash

Meta slide

These slides are already online if you want to grab them! 👇

loige.link/senior



Hello

👋 I'm Luciano

- **14 years** in the industry
- **Senior Architect** at fourTheorem
- Co-author of Node.js design Patterns
- ❤ **Node.js**, **AWS** & learning **Rust** 🦀

Let's connect

- Blog
- Twitter
- LinkedIn
- GitHub
- Twitch
- YouTube

nodejsdesignpatterns.com

A senior engineer

- The definition changes in every company
- There are many levels of seniority: senior, staff engineer, principal, etc.
- My definition: **moves projects & people forward** (*"Force multiplier"*)
- **Team player**, **not** a hero (or rockstar, superstar, magician, unicorn, etc.)

Photo by Rudolfo Spott on Unsplash

It’s not just about time or age

- ~~5 years of experience~~
- ~~At least 28 years old~~
- More time ?== more senior…

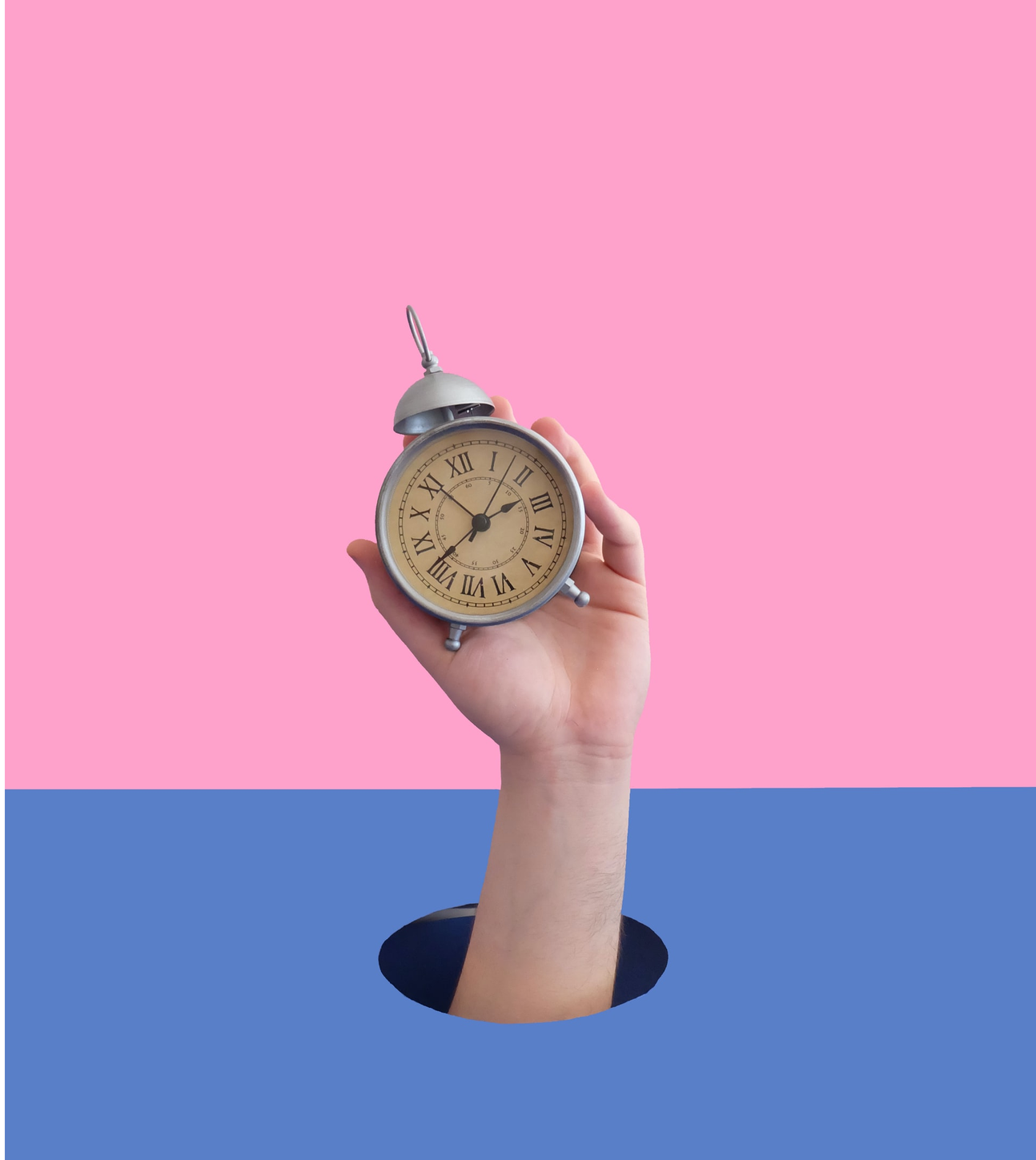


Photo by Elena Koycheva on Unsplash

Skills

Tech skills (hard skills)

🤗 Somewhat important!

Photo by Moritz Mentges on Unsplash

Soft skills

🔥 Very important!

Photo by Icons8 Team on Unsplash

Tech skills

T-shaped profile

- Master at 1 thing
- Proficient at many other things
- Example:
  - Master at **backend & API development**
  - Can do some **frontend**
  - Can do some **IaaC**
  - Understands **cloud architectures**

Photo by Lucas van Oort on Unsplash

Broad understanding

- Understand the platform
  - Architecture
  - Code Structure
  - Testing
  - Deployment process
  - Scalability model

Photo by Jen Theodore on Unsplash

Understand tradeoffs

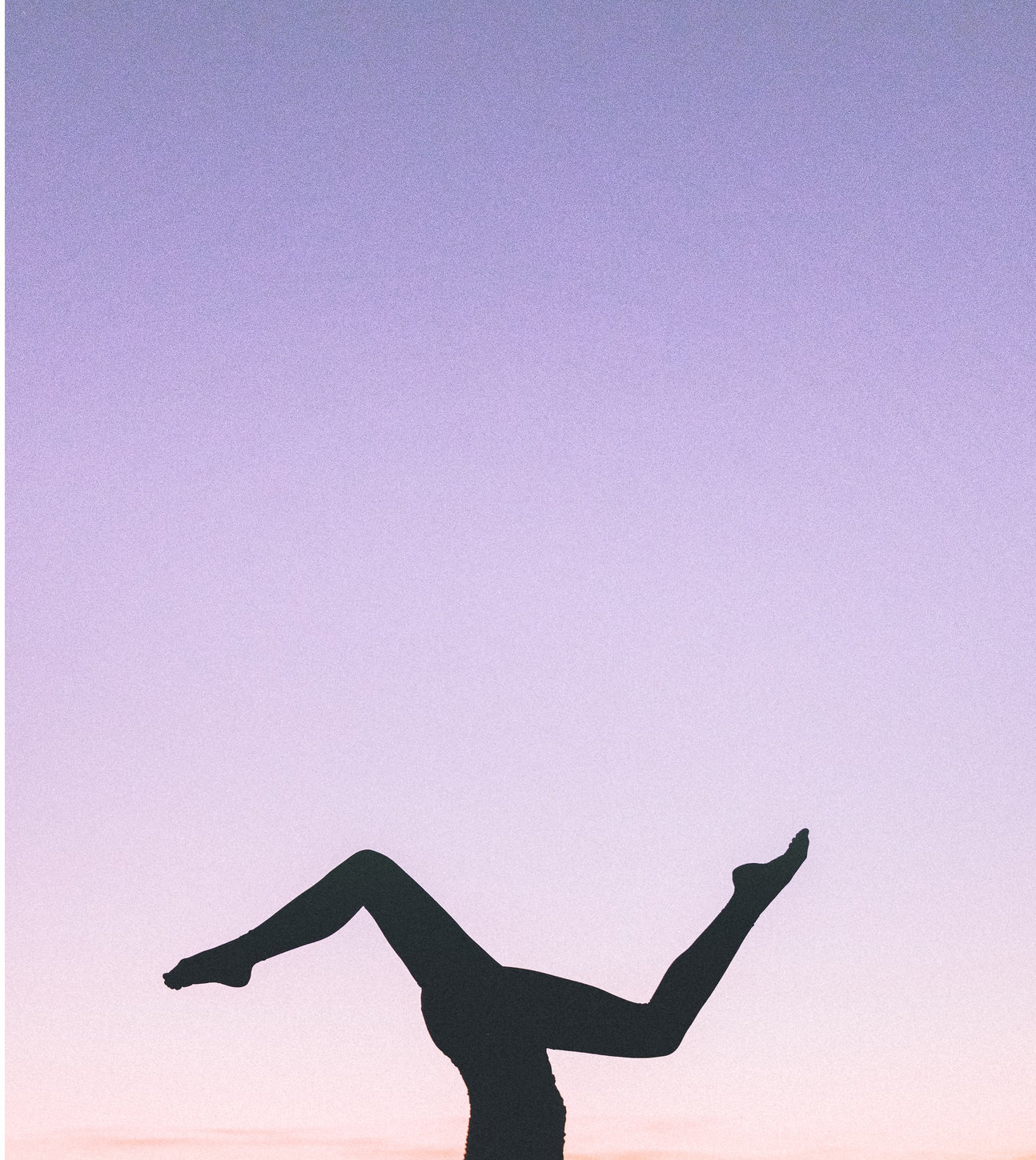
- Eg.
  - Monolith vs Microservices
  - Memory vs CPU
  - Highly Scalable vs low latency
  - Reusable vs bespoke
  - Complex (but powerful) vs Simple (but limited)
- Optimizes for the most relevant ones
- Understands the short vs long term impact of these

Photo by Pickled Stardust on Unsplash

Flexible

- Comfortable with different programming languages
- ...and paradigms:
  - OOP vs Functional
  - Declarative vs Imperative
  - Compiled vs Interpreted
- Can solve the same problem in different ways and with different tools

Photo by Wesley Tingey on Unsplash



Bug catcher

- Understands and refines user stories
- Can write different types of test:
  - Unit
  - Integration
  - E2E
- Can find and discuss edge cases
- Keeps track of **technical debt** and helps to pay it back

Photo by Benjamin Balázs on Unsplash

Good advisor

- Understand patterns and best practices
- Can suggest patterns that have good long term effects
- ...and avoids other that might lead to problems
- Suggests but does not mandate

Photo by Diego PH on Unsplash

Soft skills

Active lever

- Ask hard questions
- Takes leadership to find what they don’t know
- Is a bridge between product and technology

Understand the business

- Understand the business
  - What’s the purpose
  - What are the unique strenghts
  - What are the main weaknesses
  - How can technology help

Communication!

- Be able to talk with all the stakeholders
- Be able to talk about failures and learnings
- Can write docs & deliver presentations
- Can make complex topics digestable

Supports management

- Planning, drive cerimonies
- Can split complex tasks into manageable parts

Autonomy

- Can drive projects that require research and grind
- But aware of avoiding silos
- Knows how to get unstuck
  - Ask for help
  - Research and experiment
  - Read the docs
  - Read and understand existing code

Focus on delivery

- Set expectations
  - What does it mean to be successful in the current environment
- Positive attitude:
  - Don’t blame the system
  - Propose solutions
  - Facilitate conversations
  - Help to find compromises
- Confidence that hard problems can be solved even if we don’t know how yet


How to grow

Go 1 level deeper

- Don’t stop at the layer you are familiar with
- What happens in the underlying layers?
- E.g.
  - How does the **HTTP** protocol work?
  - How does **TCP** work?
  - **UTF-8**, **SHA512**, **DNS**, etc.
- 🎥 The computer science iceberg
- Discend one level at the time
- Build prototypes

Have fun

- Learning new stuff can be fun
- You can build utilities and side projects
- Showcase what you learned to your peers
- Can you apply these learnings at work?
- Hackaton and free study days are a great way to spend work time for engineering growth

Pair programming

- Try to pair with as many people as possible within the org
- You can probably learn something from everyone (even from the most junior)
- You can probably teach something to everyone (even to the most senior)

Create content

- Articles, Talks, Videos, Twitter threads, Etc.
- You don’t need to be an expert to share something new you learned
- Make it a habit
- Creating content will massively improve your communication skills
- ⚛ Atomic essays

Keep a positive attitude

- Be pragmatic when problems arise
- With enough time and money you can probably solve everything
- With *sligthly less* money and time you can find decent compromises 😄
- Help others
  - Especially at the early phases of a project/feature - Avoid common design mistakes early on
    - e.g. defining data models for dbs/events and keeping those documented and versioned
    - But don’t do the mistake of thinking that there’s only one solution
      - Evaluate other proposals, suggest and give people space to experiment and go ahead with other ideas
- Don’t be picky about technology or style

Additional resources

- https://www.progression.fyi/
- https://medium.com/building-carta/engineering-levels-at-carta-d33db2a55a20
- https://career-ladders.dev/engineering/
- https://davidxiang.com/2021/07/18/20-micro-habits-of-high-impact-software-engineers/

We cannot excel at everything

But we should know our strengths and weaknesses

Work with our team to amplify strenghts and compensate weaknesses

And strive to get better every day!

Thanks