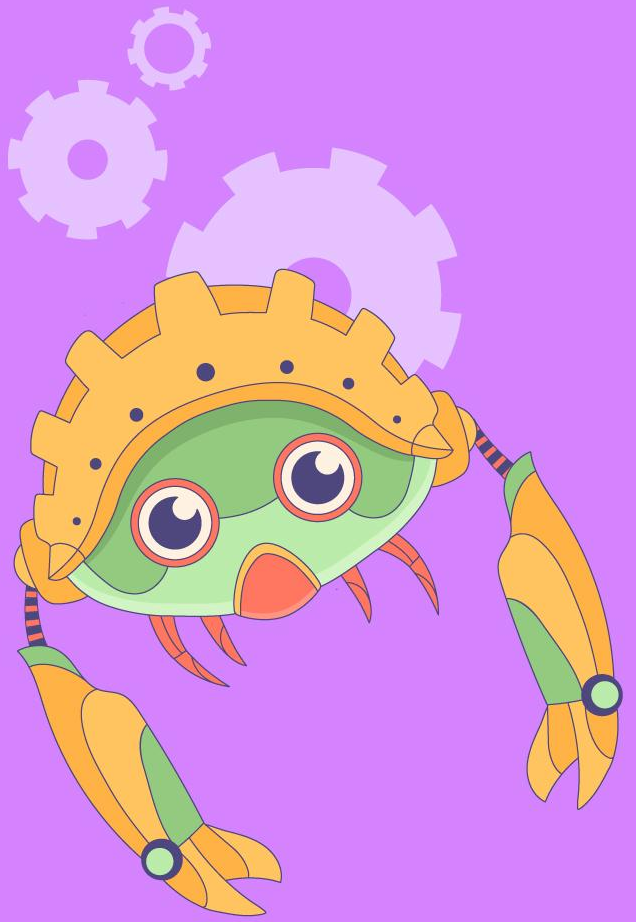
MITCHEL VROEGE

Sr. Backend Developer @ 2066d

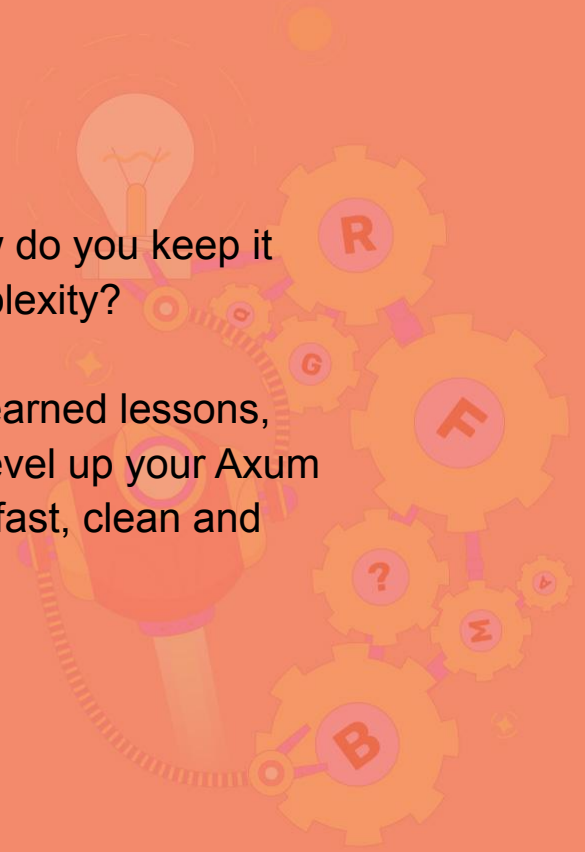
# Building Composable Web Services with Axum



# INTRODUCTION

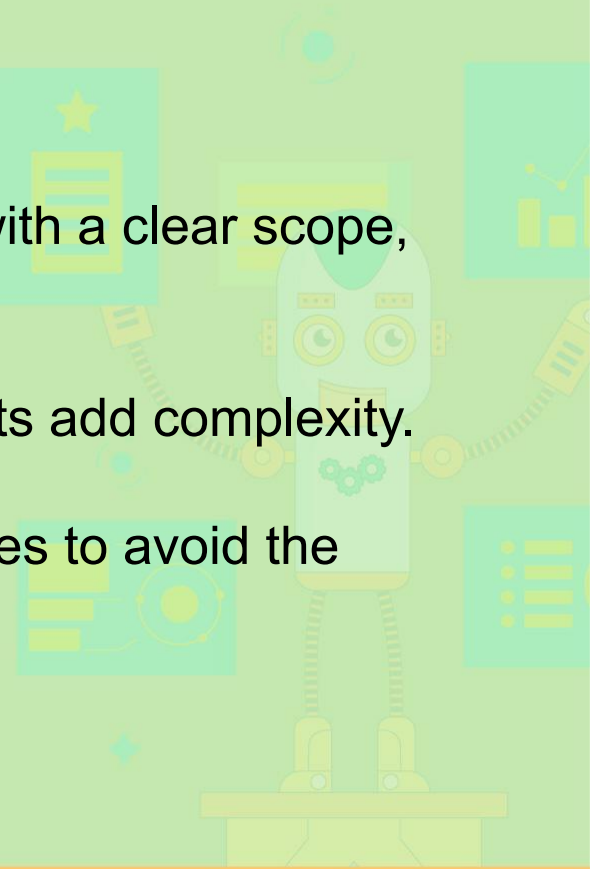
Axum makes writing a basic web service easy, but how do you keep it from getting all tangled up as it grows in size and complexity?

In this talk, we'll walk through practical patterns, hard-earned lessons, and maybe even a bad joke or two. If you're ready to level up your Axum skills, this talk will help you to keep your web backend fast, clean and maintainable, without sacrificing your sanity.



# PROBLEM

- We start small, with a clear scope, manageable.
- New requirements add complexity.
- Thoughtful choices to avoid the tangled mess.



# PURPOSE

- What do we do?
- Decide what kind of system we're building.
- Do we let it grow organically?

# PURPOSE

- Pause, and think about building with purpose.
- We're not going "enterprise"-scale here.
- To build with purpose, we need boundaries.

# PURPOSE

- Grouping models and logic in modules.
- It's not just about organizing files.
- We draw lines for independence.

# PURPOSE

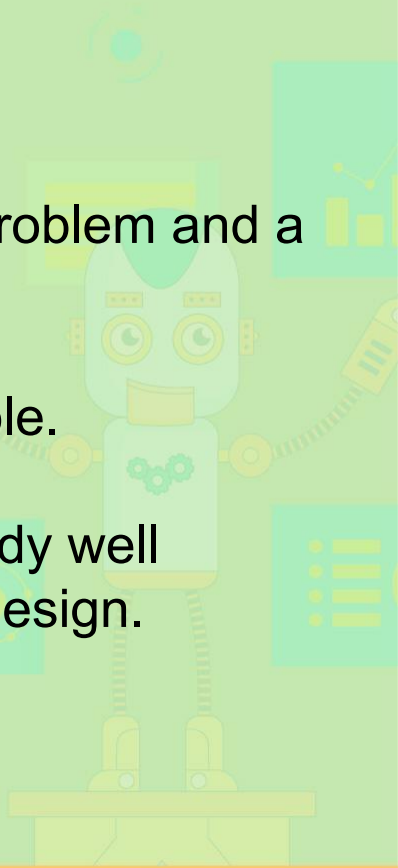
- The flow of knowledge.
- We're not eliminating complexity, we shape it.
- Later never comes.

# PURPOSE

- You only have to understand the part you're working on.
- Think ahead.
- Think ahead (yes, this is here on *purpose*, because it's important).

# PRACTICE

- We've talked about the problem and a possible solution.
- Let's make it more tangible.
- Rust and Axum are already well suited to a composable design.

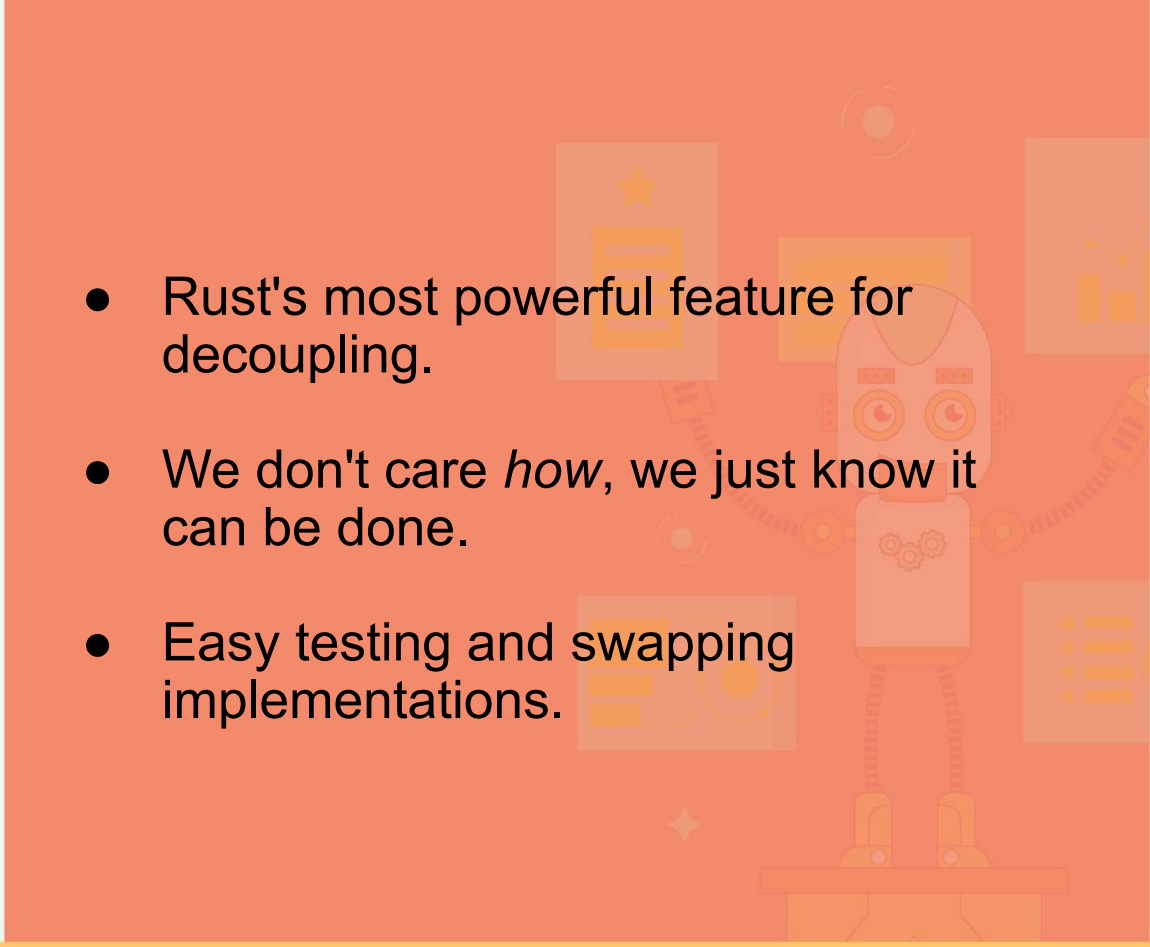


# PRACTICE

- Encapsulation.
- Make a choice about what someone is allowed to know or do with your type.
- Let the type system work *for* you.

# PRACTICE

- Rust's most powerful feature for decoupling.
- We don't care *how*, we just know it can be done.
- Easy testing and swapping implementations.

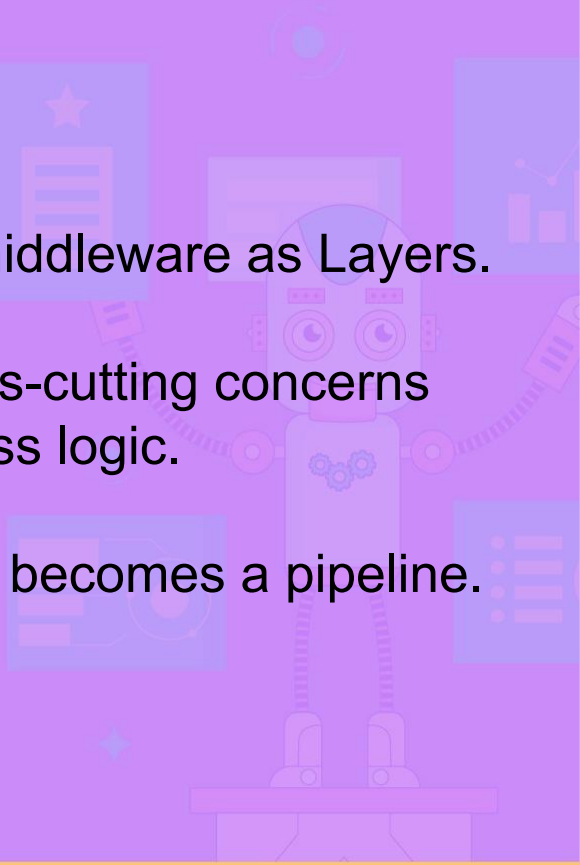


# PRACTICE

- Axum is using the full power of Rust, instead of hiding behind macros or abstractions.
- You declare what you need, Axum wires it up for you.
- Dependency management through design.

# PRACTICE

- Tower gives us middleware as Layers.
- Layers keep cross-cutting concerns out of our business logic.
- The request path becomes a pipeline.

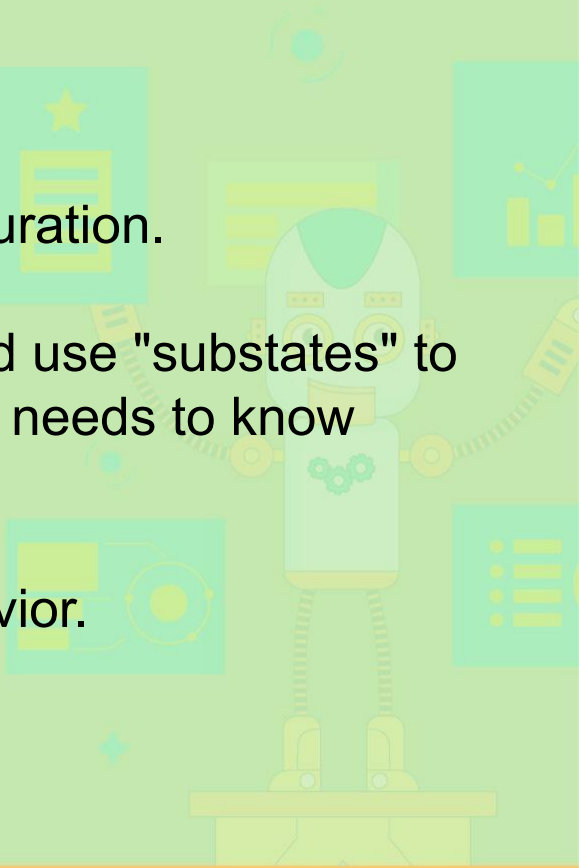


# PRACTICE

- Split your project into multiple crates with cargo workspaces.
- Each crate can be built, tested, or published independently.
- Focused pieces that are independent, but work together.

# PRACTICE

- Use typed configuration.
- Impl FromRef and use "substates" to only inject what it needs to know about.
- Predictable behavior.

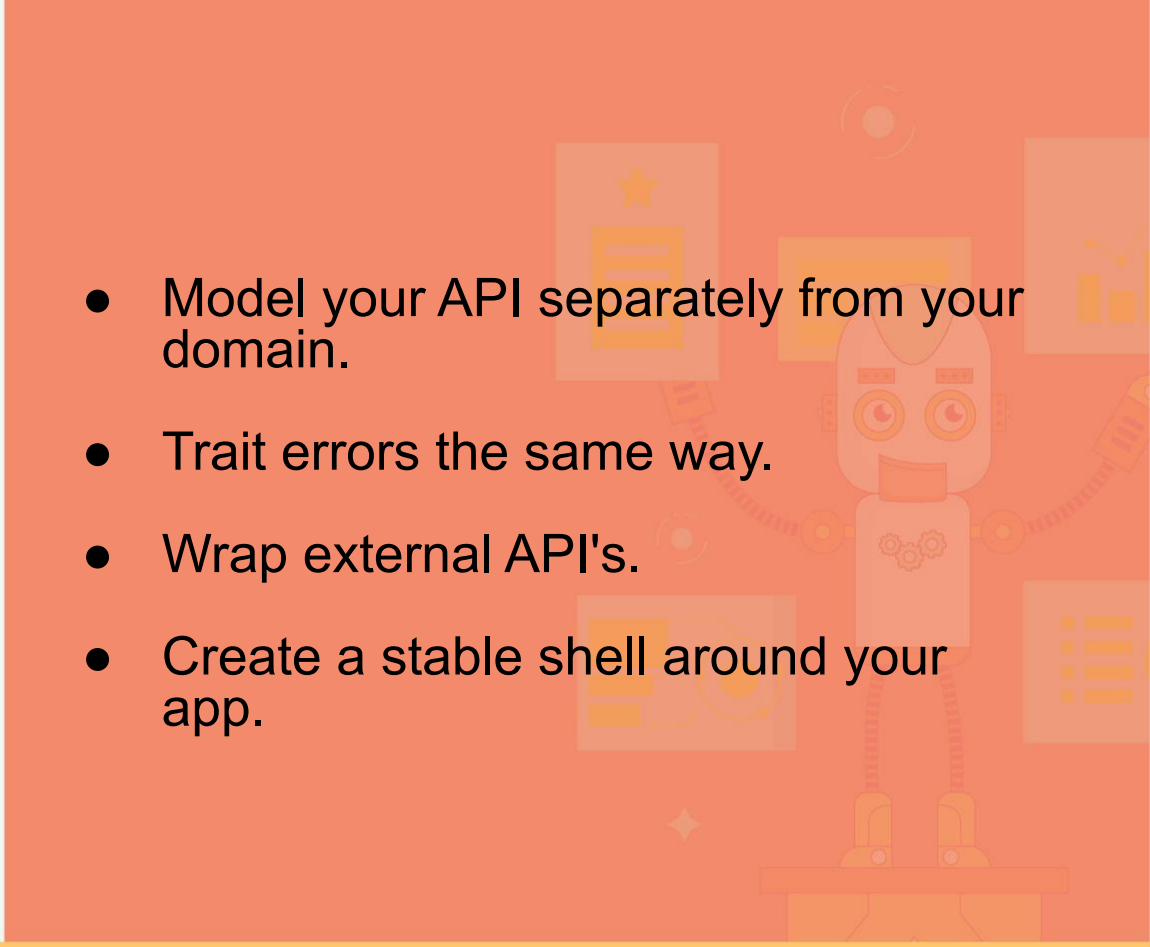


# PRACTICE

- Where the inside meets the outside.
- Every decision you make at the edge becomes "permanent".

# PRACTICE

- Model your API separately from your domain.
- Trait errors the same way.
- Wrap external API's.
- Create a stable shell around your app.

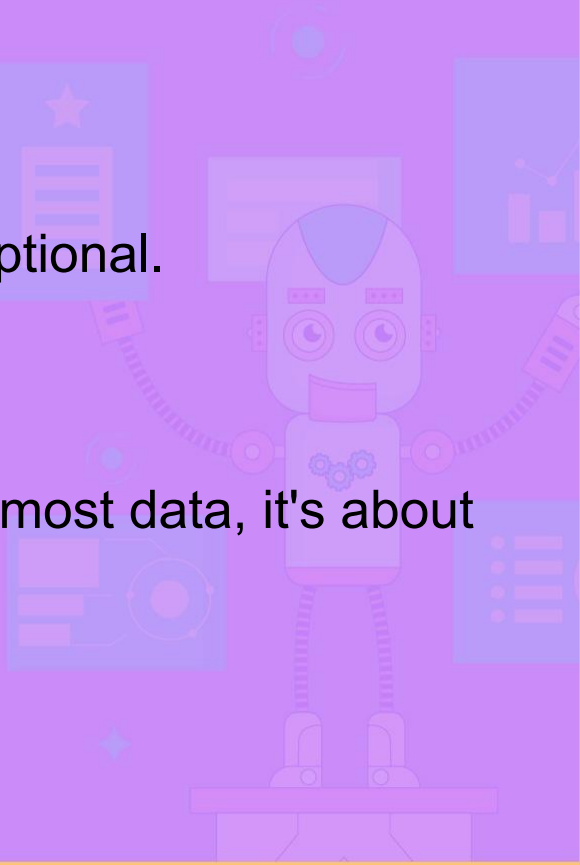


# PRACTICE

- Logic sneaking into the persistence layer.
- Keep logic in your application - loose coupling.
- Commit to a particular database - tight coupling.

# PRACTICE

- Monitoring isn't optional.
- Use tracing.
- It's not about the most data, it's about the *right* data.



# PRACTICE

- You don't need a rewrite!
- Pick one and start small.
- Improve while you build.

# PRACTICE

None of these are silver bullets. They're pieces of a bigger puzzle. Together, they give you a system that can grow in features, in people, and in time without turning into spaghetti.

And if that's not the definition of keeping your sanity, I don't know what is.

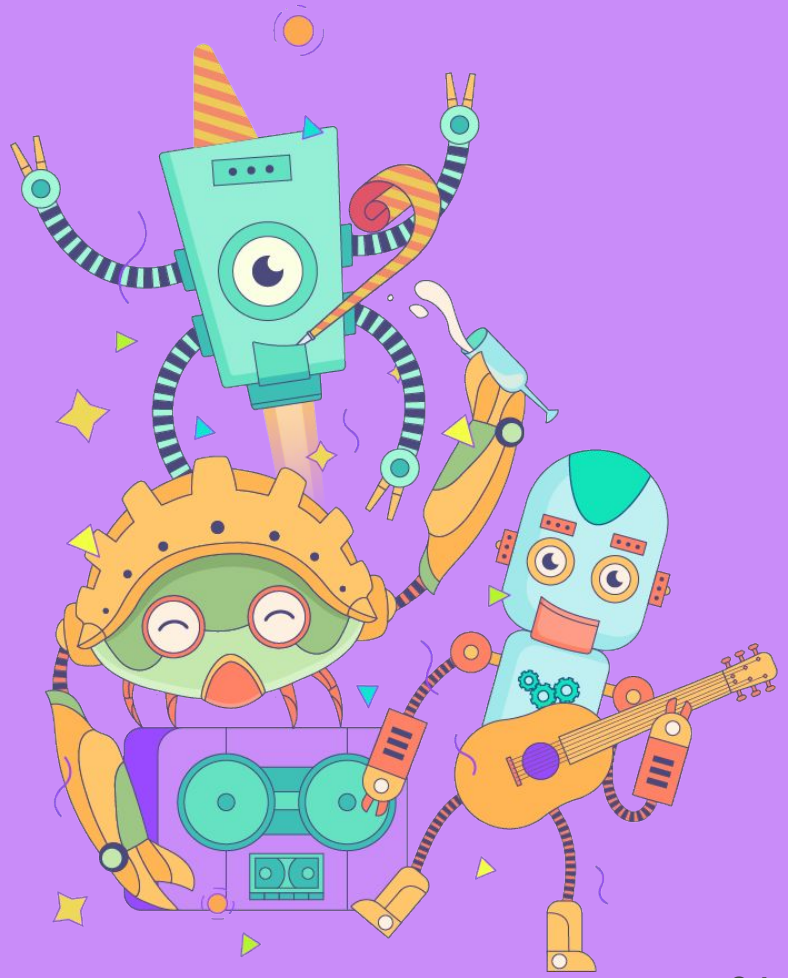
# RECAP

- We took a moment to pause.
- We looked at how the type system, traits, and modules can help shape our architecture.
- This piece does one thing, and the rest of the code doesn’t need to know how.

# RECAP

- It works for both new and existing projects.
- When you need to make a change, you know exactly how deep the water is before you dive in.
- We’re not just writing code that works today.
- Take away: less about strict rules and more about smart decisions.

# Greetings and conclusion



# MITCHEL VROEGE

mitchel@mitchelvroege.nl

# INTRODUCTION

## Subsection Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae.

## Subsection Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae.

## Subsection Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae.

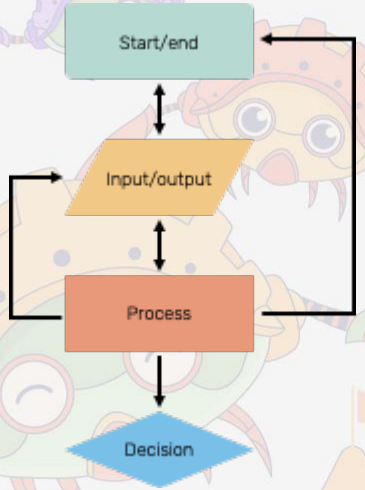
## RECAP

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae.

## Subsection Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam est ligula, faucibus nec urna sed, tristique volutpat massa. Phasellus congue leo nunc, eu vehicula enim interdum vitae. Lorem ipsum dolor sit amet, consectetur adipiscing elit.



Section Title

## Subsection Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

*Note: For use with the projector it is best to use a light theme in order to improve the accessibility and readability of the code. You can paste the code from VScode that includes the formatting and optionally adapt the background with the "Fill color" button.*

```
func hello() {
    fmt.Println("Hello world! 🌍")
}
```

Section Title

## Subsection Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

*Note: For use with the projector it is best to use a light theme in order to improve the accessibility and readability of the code. You can paste the code from VScode that includes the formatting and optionally adapt the background with the "Fill color" button.*

```
func hello() {
    fmt.Println("Hello world! 🌍")
}
```

Section Title

## Subsection Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

*Note: For use with the projector it is best to use a light theme in order to improve the accessibility and readability of the code. You can paste the code from VScode that includes the formatting and optionally adapt the background with the "Fill color" button.*

```
func hello() {
    fmt.Println("Hello world! 🌍")
}
```

## Subsection Title

```
func hello() {
    fmt.Println("Hello world! 🌍")
}
```

## Subsection Title

```
func hello() {
    fmt.Println("Hello world! 🌍")
}
```

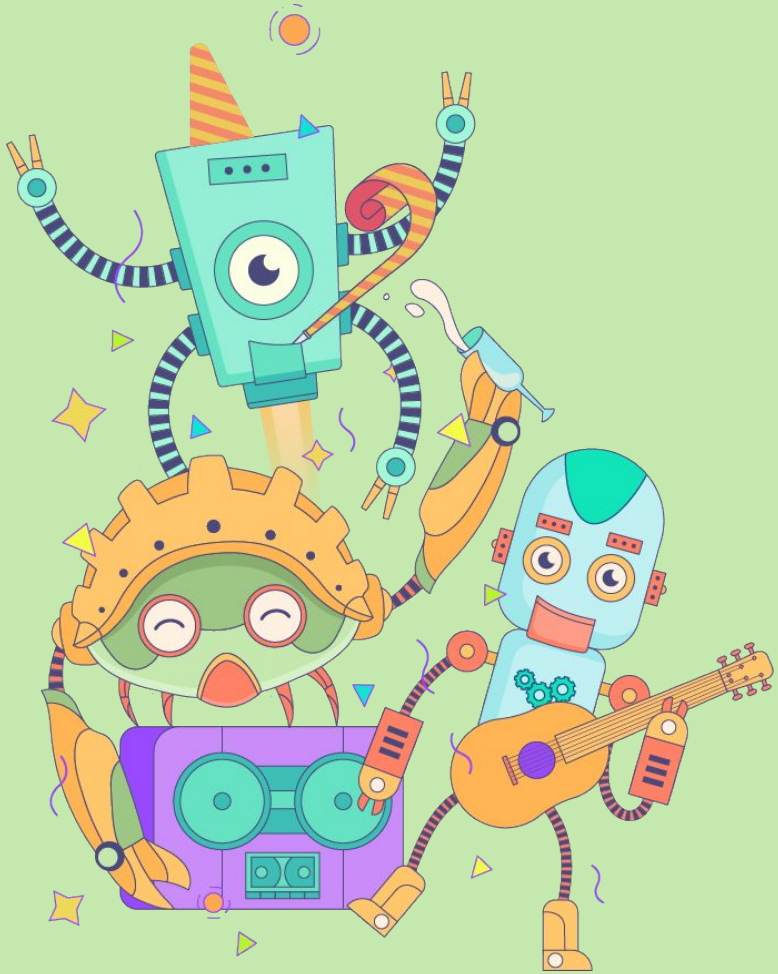
## Subsection Title

```
func hello() {
    fmt.Println("Hello world! 🌍")
}
```

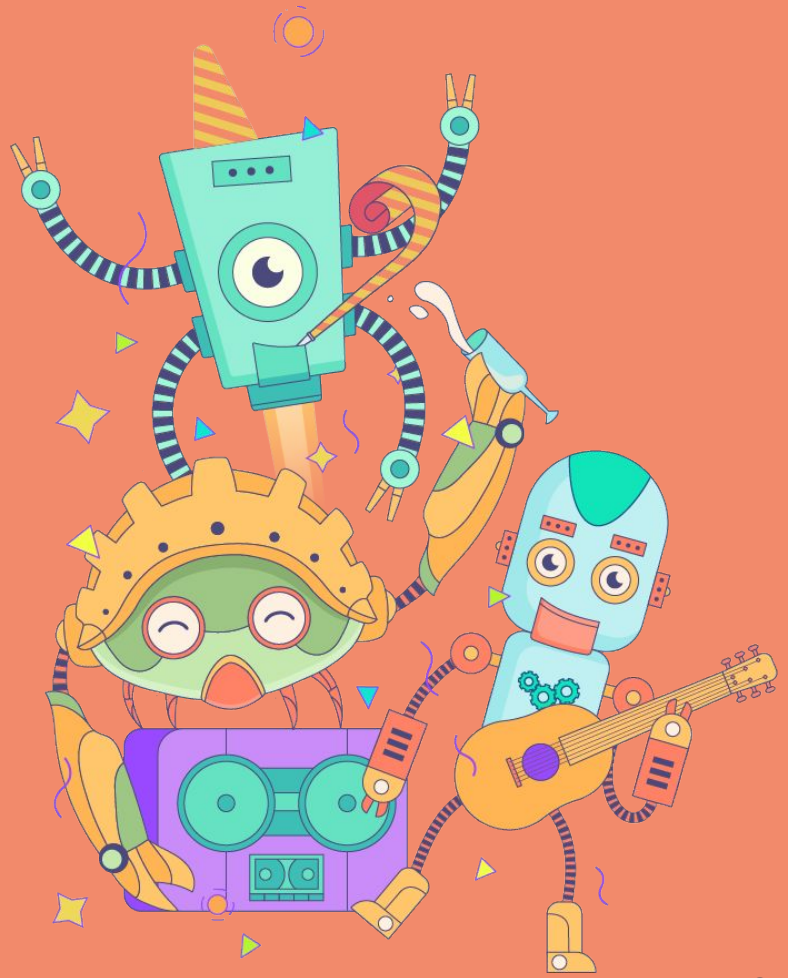
## Subsection Title

```
func hello() {
    fmt.Println("Hello world! 🌍")
}
```

# Greetings and conclusion



# Greetings and conclusion



# SPEAKER NAME

mail@speaker.com

# SPEAKER NAME

mail@speaker.com