

From Patchwork to Picasso:

Refactoring 150+ web projects and what went wrong 🚨

Muhammad Ahsan Ayaz

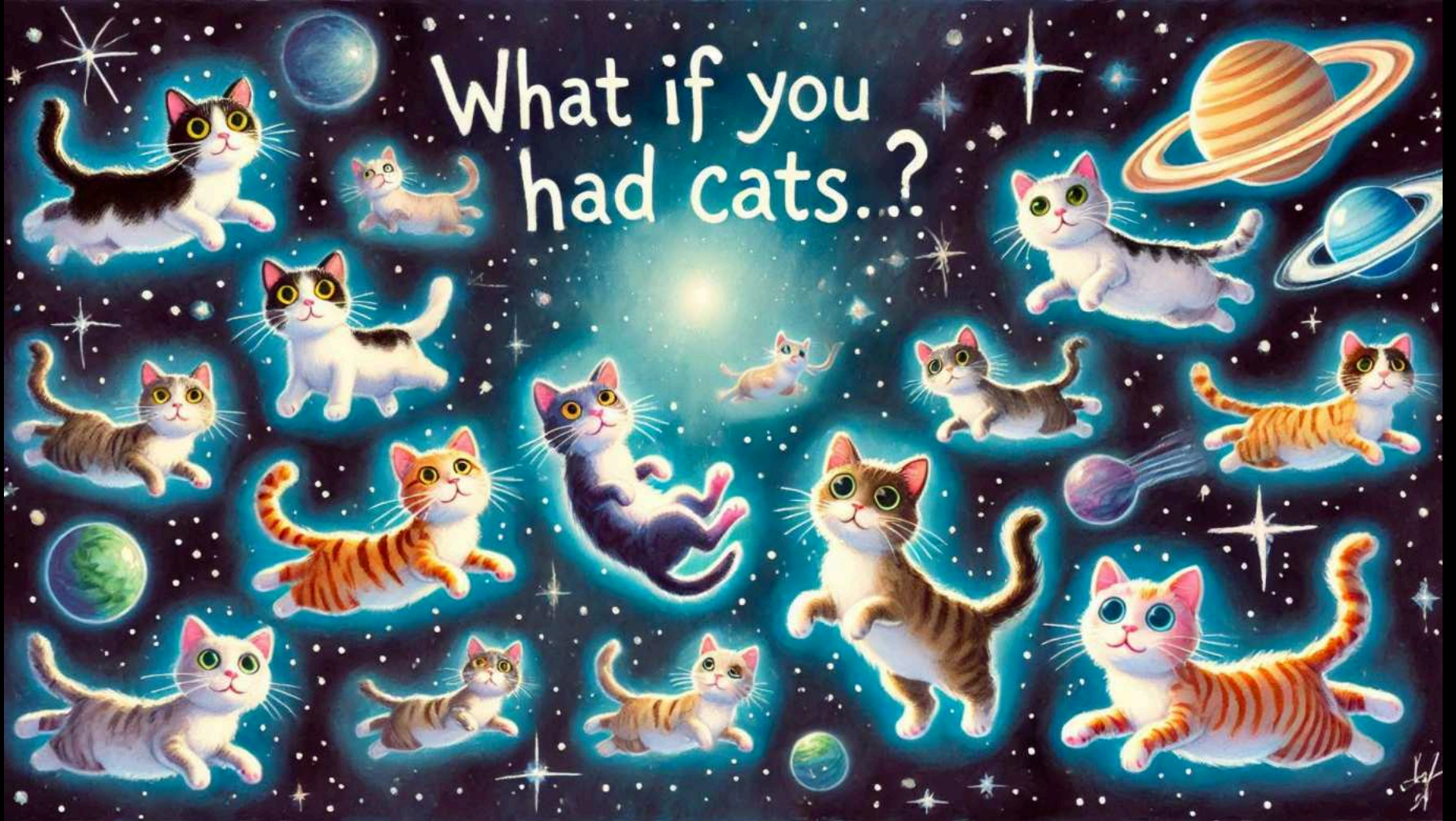
<https://codewithahsan.dev>

 @codewith_ahsan





What if you
had cats..?



150+ of them



All running in different directions while you try to walk them



That's exactly how I felt



And for the next 18-ish minutes,

And for the next 18-ish minutes,
that's what we'll talk about

Who Am I?



<https://codewithahsan.dev>

 [@codewith_ahsan](https://twitter.com/codewith_ahsan)

Who Am I?



GDE in Angular



Software Architect at Scania Group

Co-Founder at VisionWise AB and
IOMechs

<https://codewithahsan.dev>

 [@codewith_ahsan](https://twitter.com/codewith_ahsan)

Who Am I?

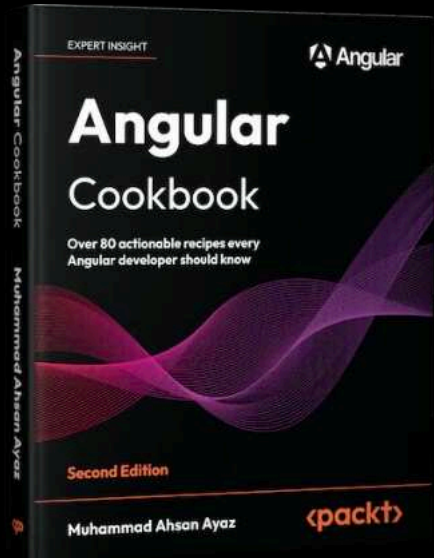


GDE in Angular



Software Architect at Scania Group

Co-Founder at VisionWise AB and
IOMechs



ng-cookbook.com

<https://codewithahsan.dev>

 [@codewith_ahsan](https://twitter.com/codewith_ahsan)



steps-to-release.md	feat(v16): migrate to v16	last year
tsconfig.json	chore: update to Angular 15	last year
tsconfig.spec.json	chore(demo): update demo to Angular 12	3 years ago
tslint.json	feat(lib): moved to angular cli generation	4 years ago

[README](#)
[MIT license](#)

[edit](#)
[menu](#)



ngx-device-detector

An Angular 6+ powered AOT compatible device detector that helps to identify browser, os and other useful information regarding the device using the app. The processing is based on user-agent. This library was built at [KoderLabs](#), which is one of the best places I've worked at ❤️

Build passing
 npm v8.0.0
 license MIT
 stars 522

[Deprecated package](#) :
 downloads 154k
 downloads 1.7k/month

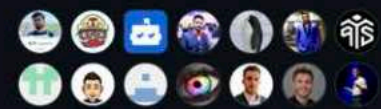
[New package](#) :
 downloads 11M
 downloads 698k/month

If you use Angular 5, you must use v1.5.2 or earlier

Used by 5.2k



Contributors 37



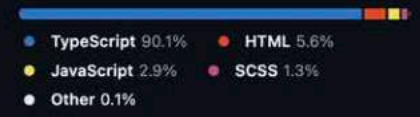
[+ 23 contributors](#)

Deployments 160

github-pages 6 months ago

[+ 159 deployments](#)

Languages











Google Cloud

2,631,014 followers

3w · Edited · 🌐

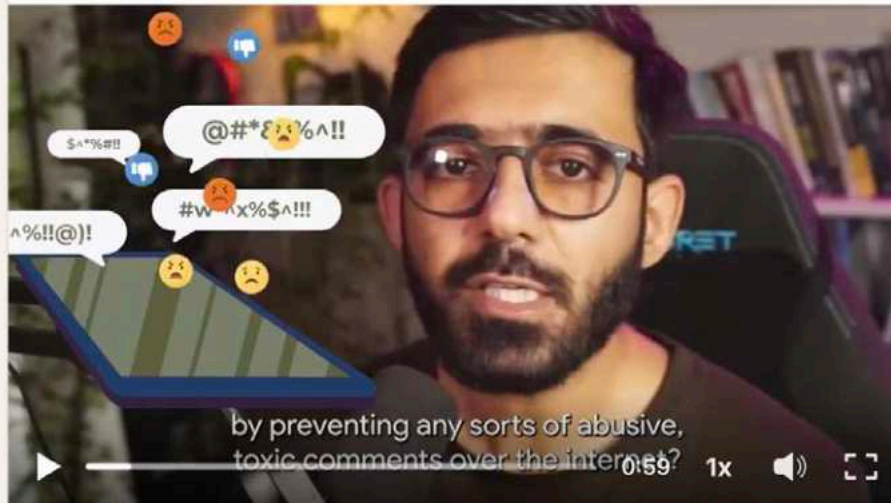
Check out this innovative sentiment analysis prototype that combines the Gemini API with Angular to enhance online safety!

Watch as developer [Muhammad Ahsan Ayaz](#) demonstrates the Gemini API's advanced capabilities:

- 🔍 Detect and prevent toxic comments in real time
- 📄 Process multiple types of content
- 🗣️ Understand extensive context for better accuracy

From sentiment analysis to user safety, developers are using the Gemini API to bring their ideas to life. Start building in AI Studio today

→ <https://goo.gle/3ANgVkl> #BuiltWithGemini



👍👎🗨️ You and 371 others

22 comments · 25 reposts

In 2019

<packt>

reached out to me for a video course

<packt>

reached out to me for a video course
on IONIC

And I thought. That's super cool



But...

I used to live with my family back then



And used to record the videos at night



I faced some challenges



I then tried to record from the living room..









Fortunately, I moved to Sweden

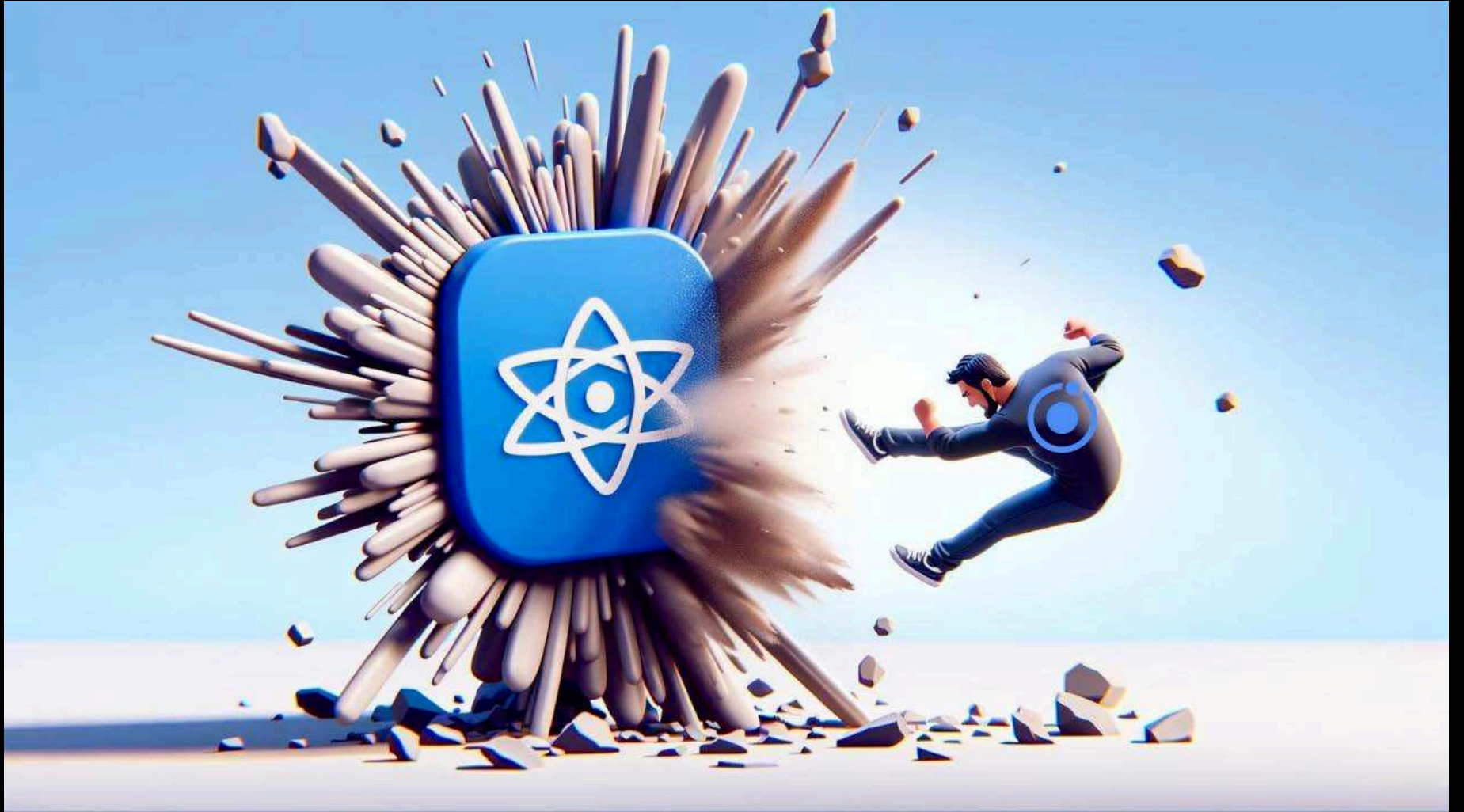


Once I published the video course, I thought...



However...

It was the time when ...

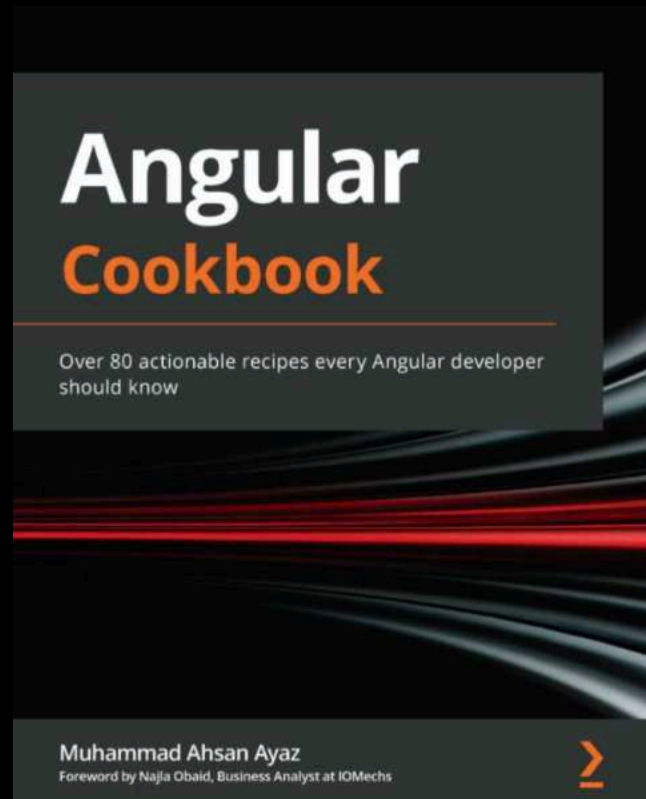


In 2020

<packt>

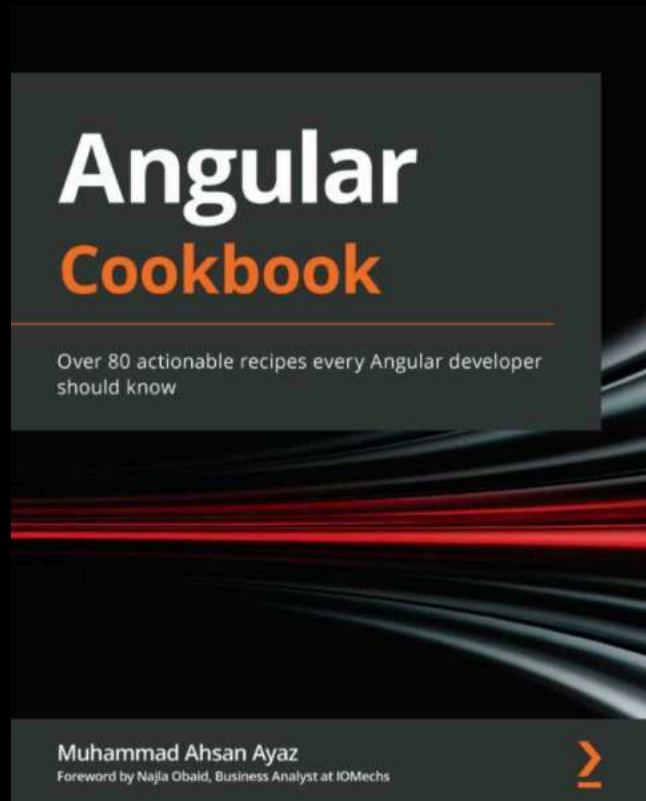
asked me if I would be interested in writing a
book on Angular

Long story short



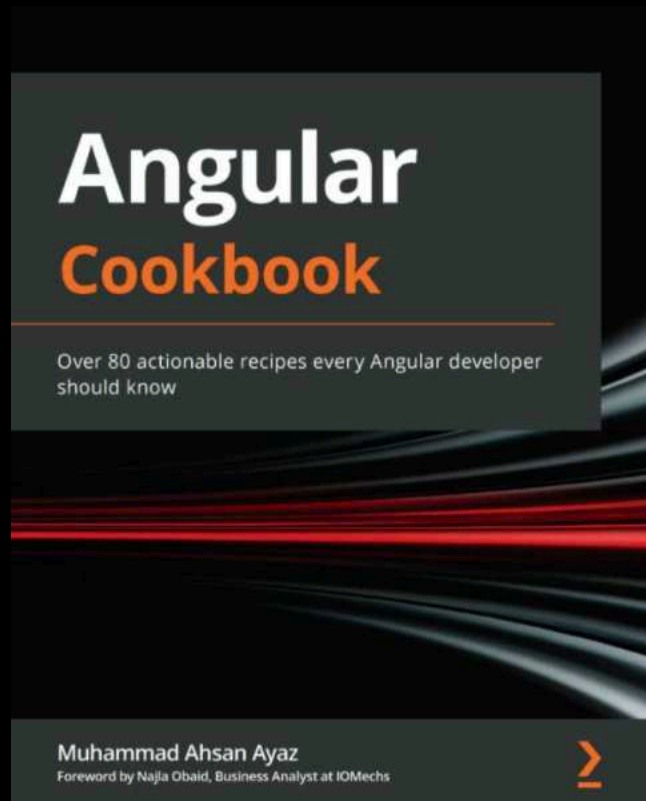
Long story short

It took over 10 months



Long story short

It took over 10 months



And we sold 2000+ copies over the years

In late 2022

<packt>

reached out for authoring the **2nd edition** of
the book

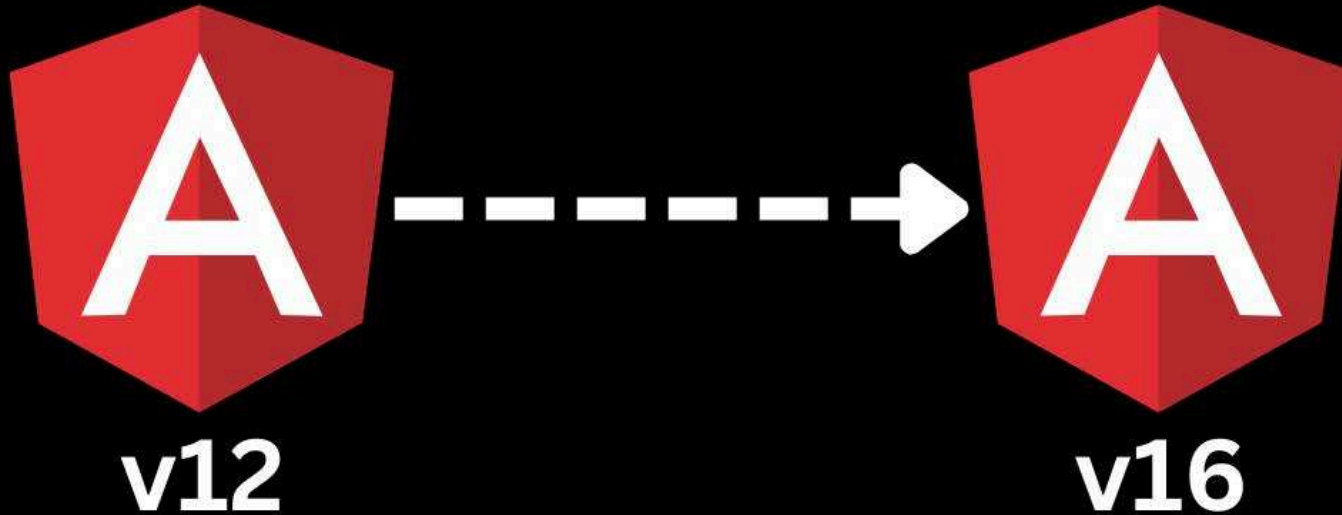
At first, I thought it would be easy

But when I looked at the code base,

But when I looked at the code base,
I knew I was in trouble

150+ projects

Waiting to be updated



Release frequency

We work toward a regular schedule of releases, so that you can plan and coordinate your updates with the continuing evolution of Angular.

✔ **HELPFUL:** Dates are offered as general guidance and are subject to change.

In general, expect the following release cycle:

- A major release every 6 months
- 1-3 minor releases for each major release
- A patch release and pre-release (`next` or `rc`) build almost every week

This cadence of releases gives eager developers access to new features as soon as they are fully developed and pass through our code review and integration testing processes, while maintaining the stability and reliability of the platform for production users that prefer to receive features after they have been validated by Google and other developers that use the pre-release builds.

Updating all the recipes

Source: <https://angular.dev/reference/releases>

Updating all the recipes



Source: <https://angular.dev/reference/releases>

So what did I do?

So what did I do?

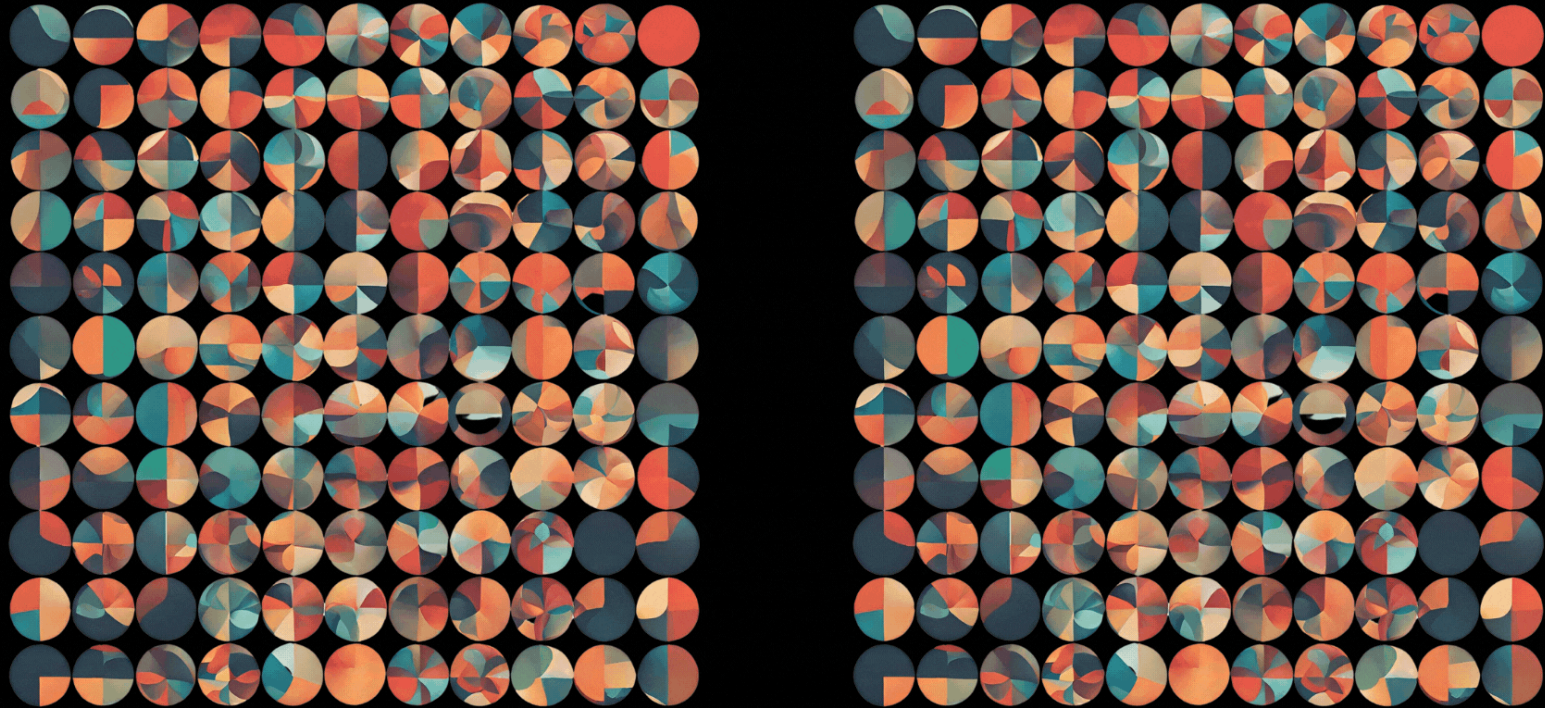


Smart Monorepos · Fast CI

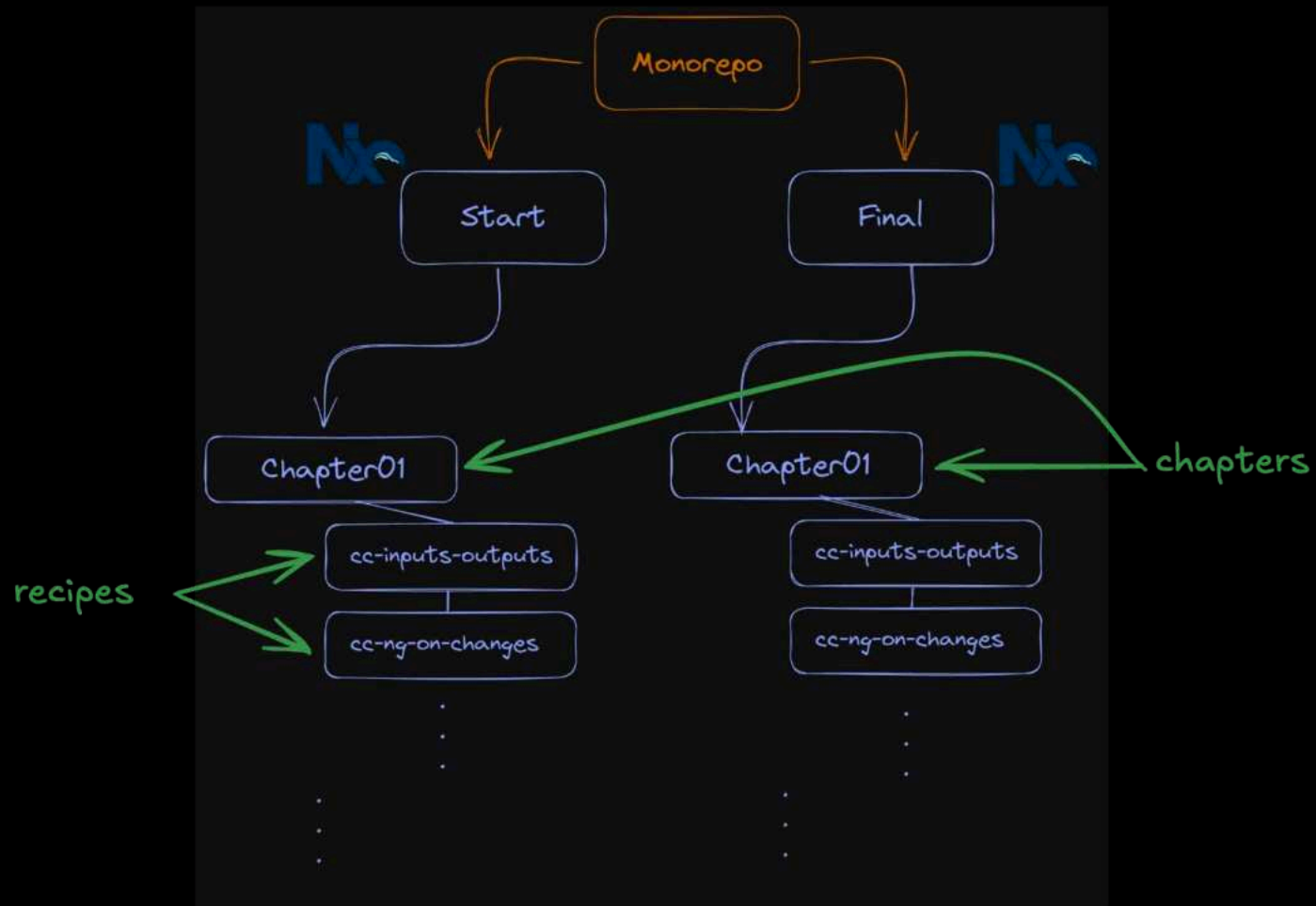
I decided to have 1 monorepo with 2 Nx workspaces.

One for the **starter** files and the other containing the **final** codes for the recipes

I decided to have 1 monorepo with 2 Nx workspaces.



One for the **starter** files and the other containing the **final** codes for the recipes





- How many of you love UI/UX?

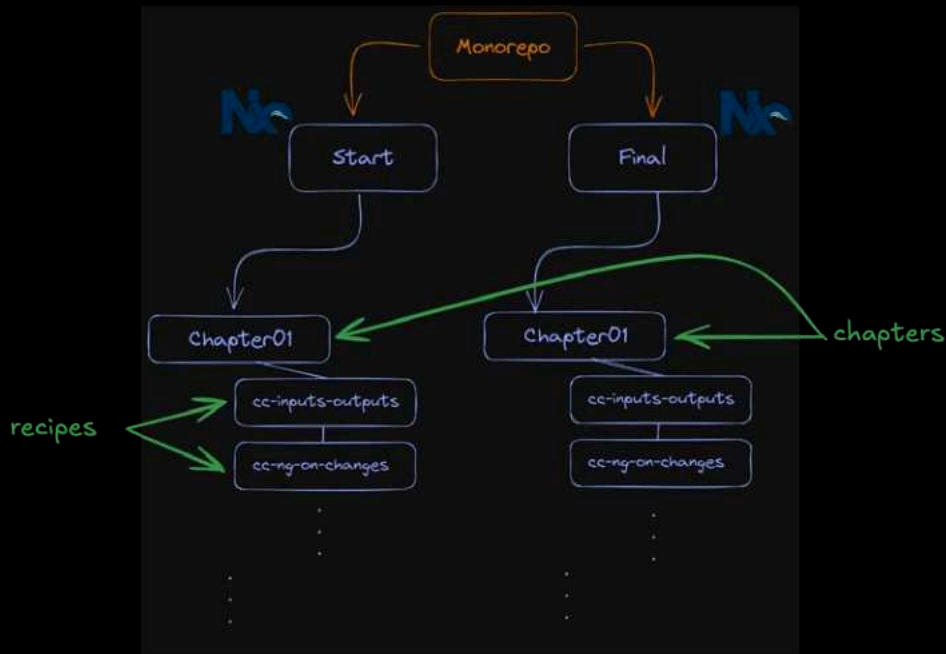
-

-

- How many of you love UI/UX?
- How many of you advocate for great UI/UX?
-

- How many of you love UI/UX?
- How many of you advocate for great UI/UX?
- How many of you advocate for great **Developer Experience (DX)**?

With the current NX structure



```
ANGULAR-COOKBOOK-2E
├── .github
├── .nx
├── .vscode
├── codewithahsan
├── dist-app
├── docs
├── final
├── node_modules
├── scripts
├── snippets
├── start
├── .editorconfig
├── .gitignore
├── .prettierrc
└── LICENSE
```

To run any recipe, my readers would
have to run:

To run any recipe, my readers would have to run:

```
cd start
nx serve chapter01-cc-ng-on-changes
# OR
cd start
nx serve chapter01-cc-inputs-outputs
# OR
cd start
nx serve chapter-09-ng-cdk-virtual-scroll
```

As developers, we want to be

As developers, we want to be
DRY

As developers, we want to be
DRY



But also remember, we need a really
good **DX**

But also remember, we need a really
good **DX**

Developer Experience

The first goal was to enable the readers to do the following:

```
npm run serve chapter01-cc-inputs-outputs  
# OR  
npm run serve chapter01-cc-inputs-outputs final
```

The second goal was to avoid typing the chapter name when running a recipe

```
npm run serve chapter01-cc-inputs-outputs # ❌
```

```
npm run serve cc-inputs-outputs # ✅
```

For the first goal, I wrote a shell script

serve-projects.sh

```
cd "start" && npx nx serve "$APP_NAME" --port=4200 -o
```

package.json

```
{  
  ...,  
  "scripts": {  
    "serve": "sh scripts/serve-projects.sh",  
    ...  
  }  
}
```

With this change, I could now simply run from my workspace
root

```
npm run serve chapter01-cc-inputs-outputs
```

OR

```
# runs the app from the `final` NX workspace  
npm run serve chapter01-cc-inputs-outputs final
```

Now, I had to focus on our second goal

```
npm run serve chapter01-cc-inputs-outputs # ❌
```

```
# avoid having to type the chapter name
```

```
npm run serve cc-inputs-outputs # ✅
```

To avoid using the chapter name, I had three possibilities when creating new projects:

-
-
-

To avoid using the chapter name, I had three possibilities when creating new projects:

- Excluding the chapter name **before** it is created
-
-

To avoid using the chapter name, I had three possibilities when creating new projects:

- Excluding the chapter name **before** it is created
- Renaming the project **after** it is created
-

To avoid using the chapter name, I had three possibilities when creating new projects:

- Excluding the chapter name **before** it is created
- Renaming the project **after** it is created
- Give up and start writing a **PhP** book instead

I chose the 2nd approach

I chose the 2nd approach


Why??

©bombayssoftwares



I don't know

Coming back to the problem at hand, to be able to do this...

```
# avoid having to type the chapter name  
npm run serve cc-inputs-outputs # 
```

I created an NX plugin using the
@nrwl/nx-plugin package

Why??

Let's have a look at how I was creating the applications for
my book then:

Let's have a look at how I was creating the applications for my book then:

```
cd start && npx nx g @nrwl/angular:application chapter01/cc-ng-on-changes
```

Let's have a look at how I was creating the applications for my book then:

```
cd start && npx nx g @nrwl/angular:application chapter01/cc-ng-on-changes
```

The above generates an app with the name

Let's have a look at how I was creating the applications for my book then:

```
cd start && npx nx g @nrwl/angular:application chapter01/cc-ng-on-changes
```

The above generates an app with the name

chapter01-cc-ng-on-changes

chapter01-cc-ng-on-changes



The plugin I created contained two counterparts



1



cc-inputs-outputs

 Plugin

 Generator



- generates app
- adds rename executor

- src
- tsconfig.json
- **project.json**
- ...



```
const addRenameScript = (  
  config: ProjectConfiguration,  
  options: RenameScriptOptions  
) : ProjectConfiguration => {  
  return {  
    ...config,  
    targets: {  
      ...config.targets,  
      rename: {  
        executor: '@codewithahsan/ng-cookbook-recipe:rename',  
        options: {  
          chapter: options.chapterName,  
          app: options.recipeName,  
        },  
      },  
    },  
    tags: options.tags,  
  };  
}
```

Executor Script

Target recipe name

```
// start/apps/chapter01/ng-on-changes/project.json
"rename": {
  "executor": "@codewithahsan/ng-cookbook-recipe:rename",
  "options": {
    "chapter": "chapter01",
    "app": "cc-ng-on-changes"
  }
}
```


2



~~chapter01-cc-inputs-outputs~~

 Plugin

 Executor

- reads project.json
- renames the app name

- src
- tsconfig.json
- **project.json**
- ...

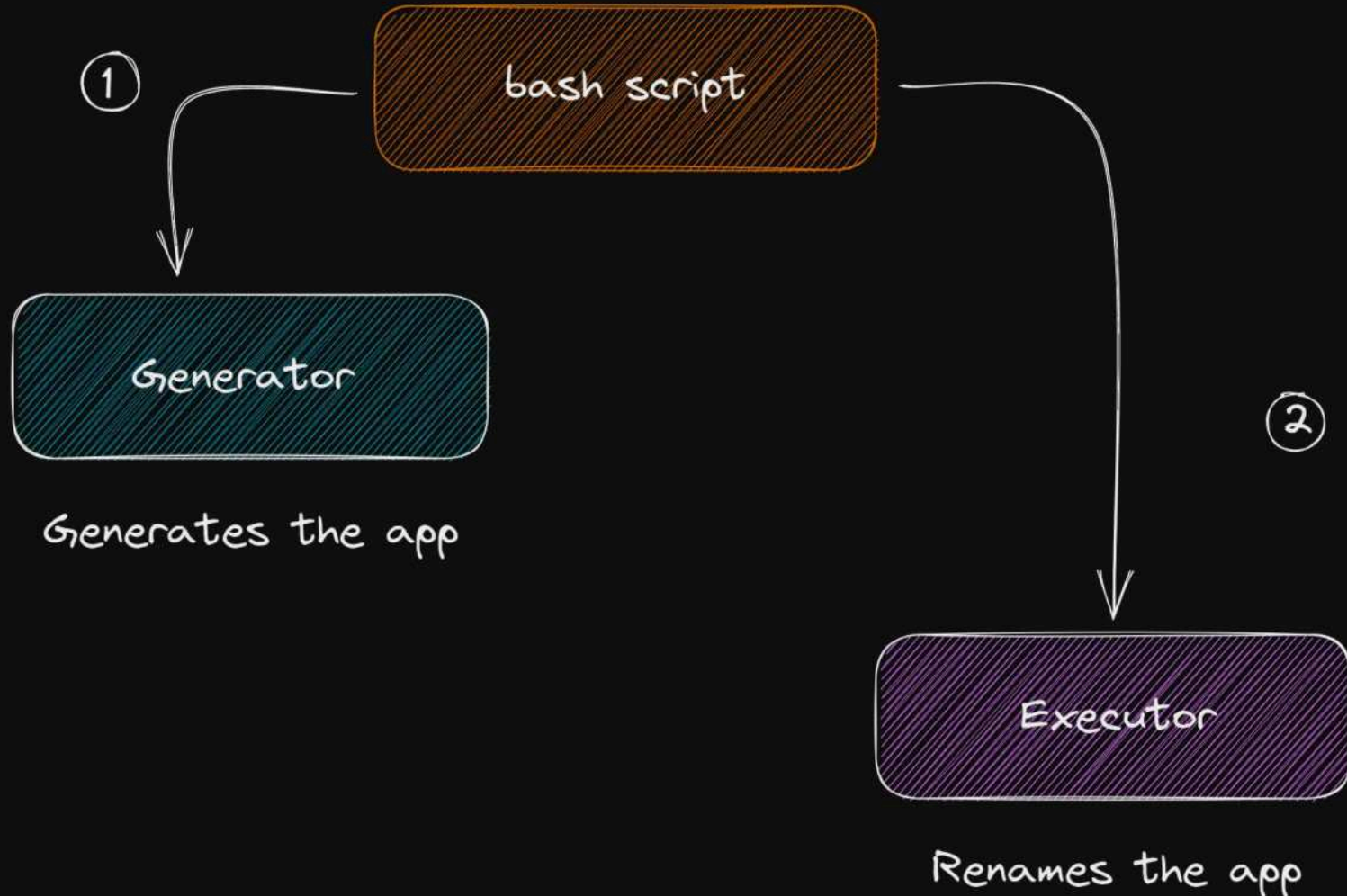
cc-inputs-outputs

@codewithahsan/.../executors/rename/executor.ts

```
const renameApp = async (
  options: BuildExecutorSchema,
  context: ExecutorContext
) => {
  const { chapter, app: appName } = options;
  if (!chapter) {
    return;
  }
  console.log('Executor running for Build', { chapter, appName });
  const projectJSONFilePath =
    `${context.root}/apps/${chapter}/${appName}/project.json`;
  const { originalName } = await updateProjectJSON(
    projectJSONFilePath,
    appName
  );
  if (!originalName.endsWith('e2e')) {
    const jestConfigFilePath =
      `${context.root}/apps/${chapter}/${appName}/jest.config.ts`;
    try {
      await updateJestConfig(jestConfigFilePath, appName, originalName);
    } catch (err) {
      console.log('error caught while finding jest file', err);
    }
  }
  console.log(`Renaming project done. New name = ${appName}`);
};
```

Which means, if I run:

```
npm run create 01 cc-ng-on-changes "NG On Changes"
```



Let's dissect the command to understand better

package.json script
& bash script

chapter number

app name

```
npm run create 01 cc-ng-on-changes "Component  
Communication using ngOnChanges"
```

App's Title (to show in the header)

And this is the bash script

```
# code stripped for conciseness

CHAPTER="chapter$1"
APP_NAME=$2
APP_FULL_NAME=`echo "$CHAPTER/$APP_NAME" | sed -r 's/[//]+/-/g'`
APP_TITLE=$3

echo "creating project for start"
cd "start" && npx nx g @codewithahsan/ng-cookbook-recipe:ng-cookbook-recipe
"$APP_NAME" --title="$APP_TITLE" --directory="$CHAPTER" --style scss\
--routing --e2eTestRunner none --skipDefaultProject --addTailwind

npx nx run "$APP_FULL_NAME:rename"
```


I could now run:

I could now run:

```
npm run serve cc-ng-on-changes
```

Do you remember what our goal was?

Do you remember what our goal was?

A great DX...

Do you remember what our goal was?

A great DX...



I think we achieved it, at least to some
extent

And to be honest, I was...





schematics

And just as I felt enlightened...

And just as I felt enlightened...

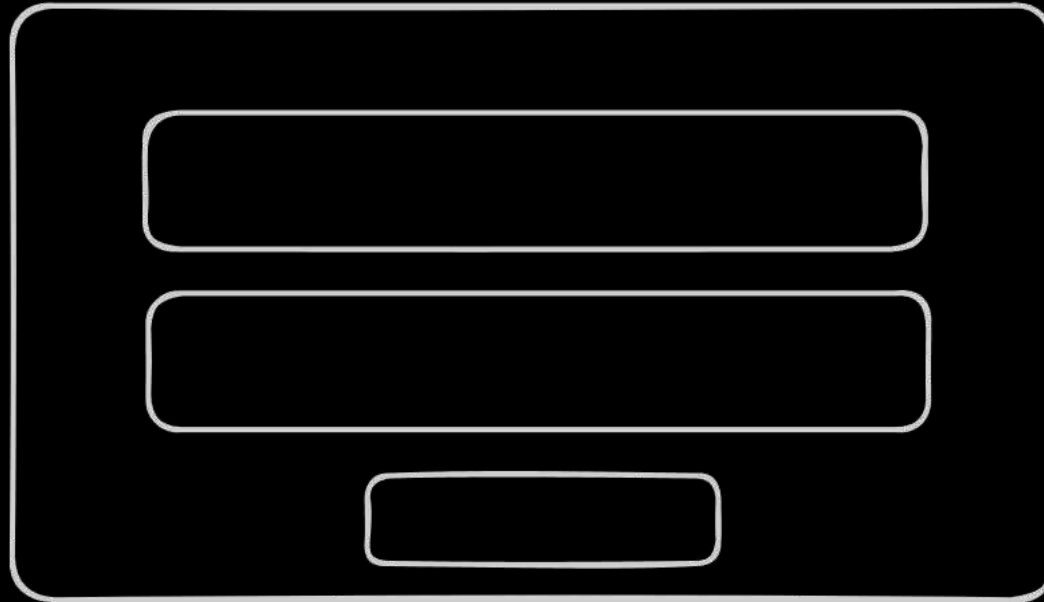


Have you heard this term called
consistency?

The idea was to have the same header across all the projects

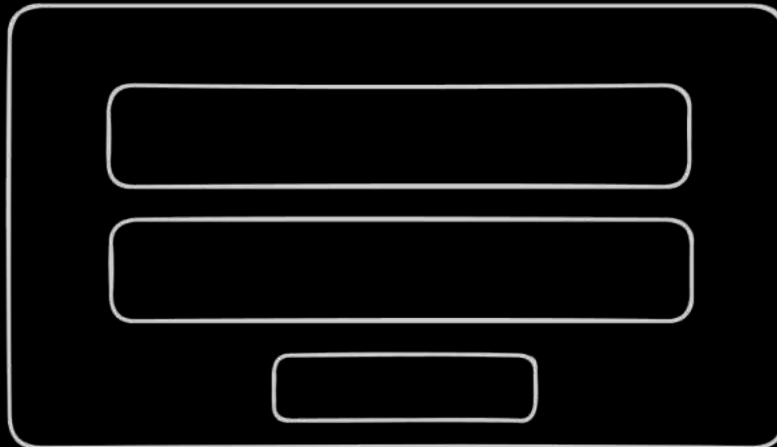


Angular Signals





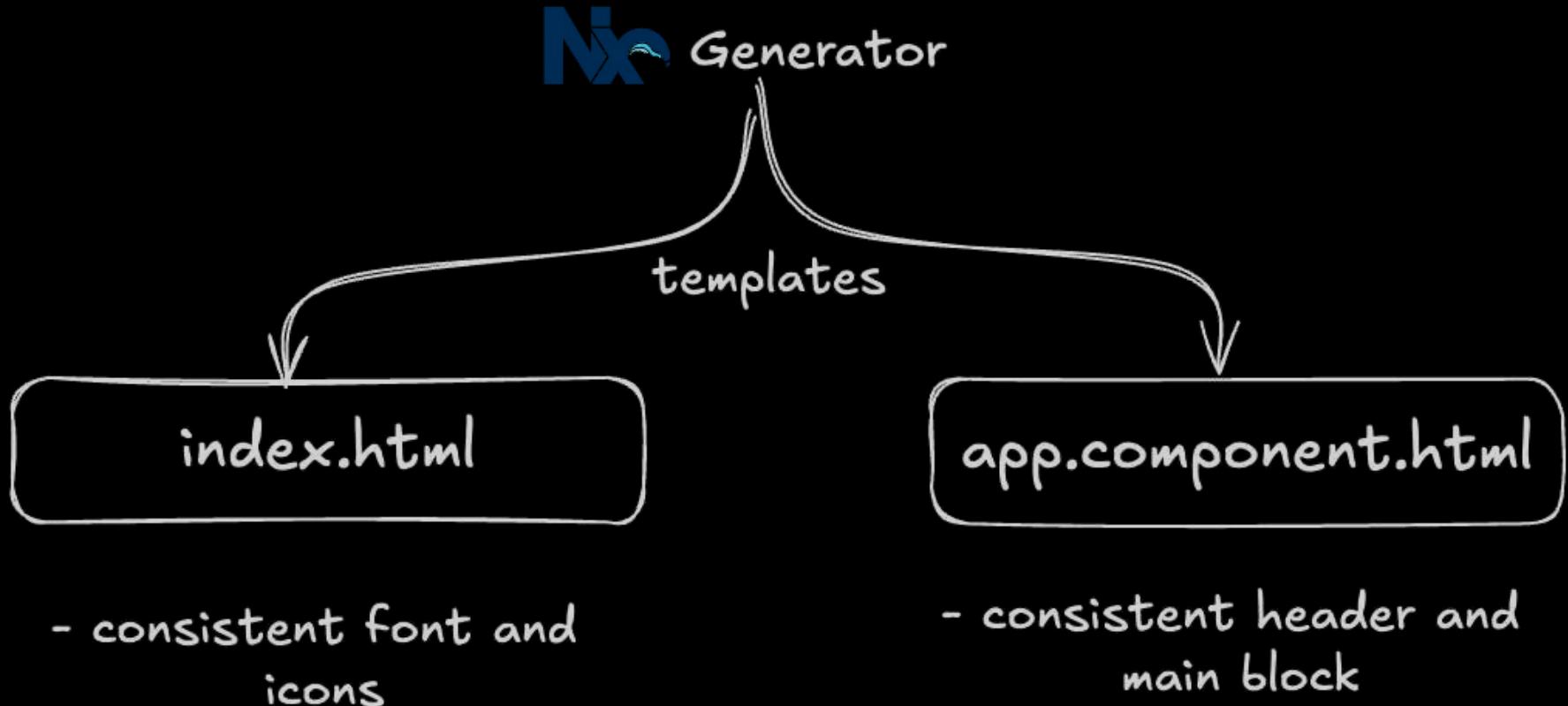
Angular Signals



Dynamic
Main Content

Same Header
in all recipes

I thought I could do the following:



index.html__template__

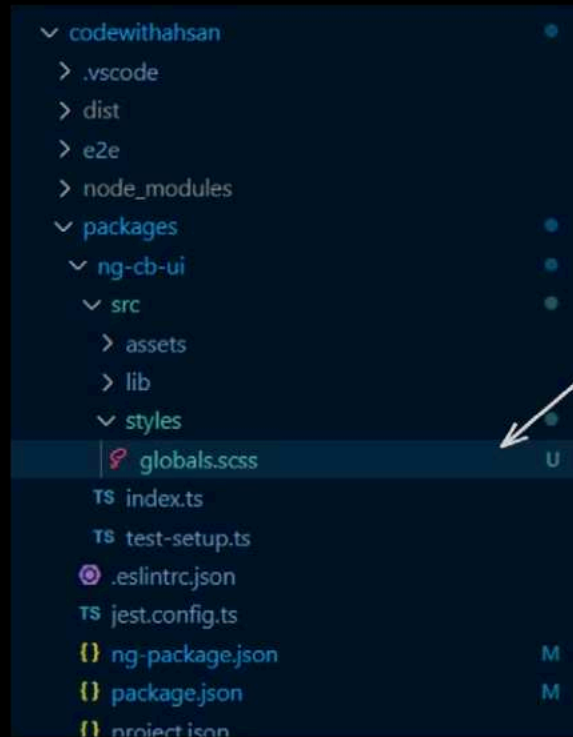
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title><%= title %></title>
    <base href="/" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?
family=Nunito:wght@400;500;600;700&display=swap" rel="stylesheet">
    <link rel="stylesheet" href="https://fonts.googleapis.com/css2?
family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50.
.200" />
    <link rel="icon" type="image/x-icon" href="favicon.ico" />
  </head>
  <body>
    <<%= prefix %>-root></<%= prefix %>-root>
  </body>
</html>
```


But we only discussed HTML templates here. What about the biggest fear of web developers?



Well, this is when I created an NX
Library.

Nx A Library



used in all recipes



Nx A Library

```
ng-package.json M X
codewithahsan > packages > ng-cb-ui > ng-package.json > [ ] assets
You, 1 minute ago | 1 author (You)
1 {
2   "$schema": "../..../node_modules/ng-packagr/ng-package.schema.json",
3   "dest": "../..../dist/packages/ng-cb-ui",
4   "lib": {
5     "entryFile": "src/index.ts"
6   },
7   "assets": [
8     { "input": "src/styles", "glob": "**/*.scss", "output": "styles" },
9     { "input": "src/assets", "glob": "**/*", "output": "assets" }
10  ]
11 }
12
```

packaged by
ng-packager

```
package.json M X
codewithahsan > packages > ng-cb-ui > package.json > ( ) exports > ( ) ./styles/globals
You, 40 seconds ago | 1 author (You)
1 {
2   "name": "@codewithahsan/ng-cb-ui",
3   "version": "0.0.1",
4   "peerDependencies": {
5     "@angular/common": "^16.0.0",
6     "@angular/core": "^16.0.0"
7   },
8   "dependencies": {
9     "tslib": "^2.3.0"
10  },
11  "sideEffects": false,
12  "exports": {
13    "./styles/globals": {
14      "sass": "./styles/globals.scss"
15    }
16  }
17 }
18
```

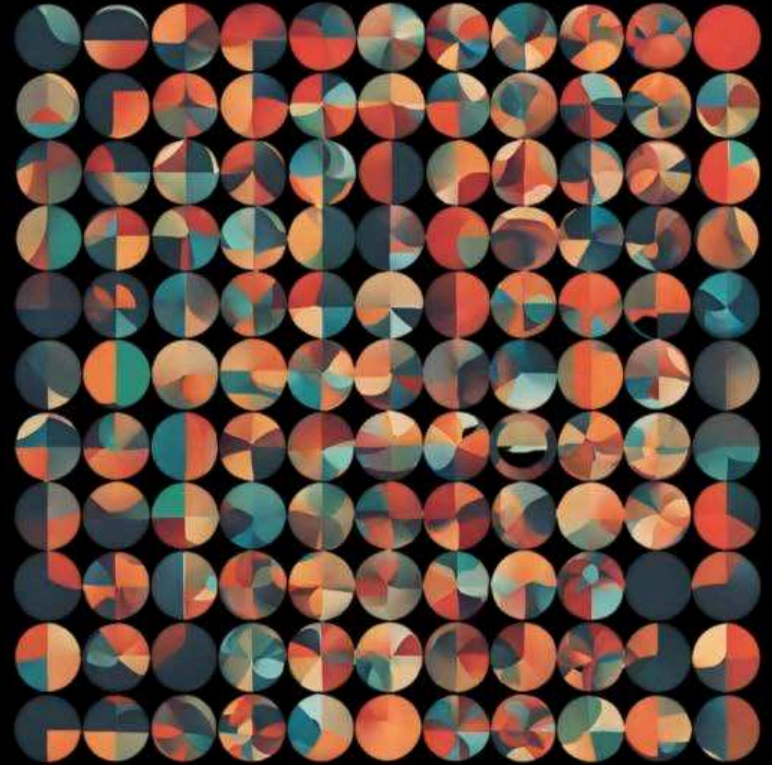
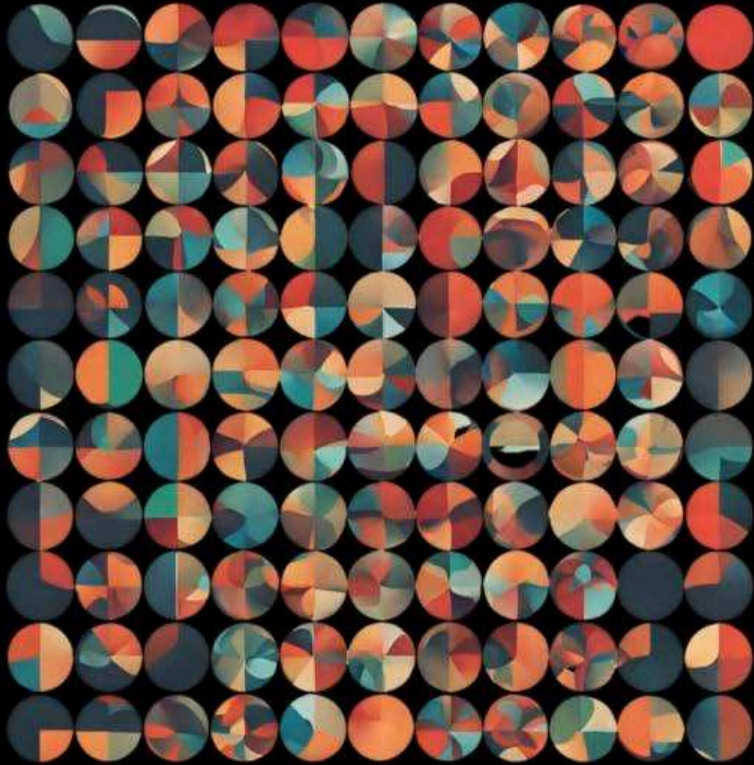
exported by
package.json

So I added another function in my



```
export const addStyles = (
  config: ProjectConfiguration
): ProjectConfiguration => {
  return {
    ...config,
    targets: {
      ...config.targets,
      build: {
        ...config.targets.build,
        options: {
          ...config.targets.build.options,
          styles: [
            ...config.targets.build.options.styles,
            'node_modules/@codewithahsan/ng-cb-ui/styles/globals.scss',
          ],
        },
      },
    },
  };
}
```

Adds the styles automatically to newly generated recipes



And the result was... 🥁

Final Output



(2 / 3)

Add

- Buy milk
- Read a book
- Delete PHP from existence

All

Active

Completed

And just as I thought everything was



and



And just as I thought everything was



Something **BIG** happened!

Can you guess??

Angular Changed their Logo 🤖

Angular Changed their Logo 🤖



And now I was left with 150+ projects having the hardcoded
Angular logo (from the generator)




I composed myself!



To this:

app.component.html__template__



```
<ng-cb-ui-header  
  appTitle="<%= title %>"  
  appName="<%= recipeName %>">  
</ng-cb-ui-header>  
  
<main class="content" role="main">  
  <!-- Your code here -->  
</main>
```

Does that sound good?? 🙋

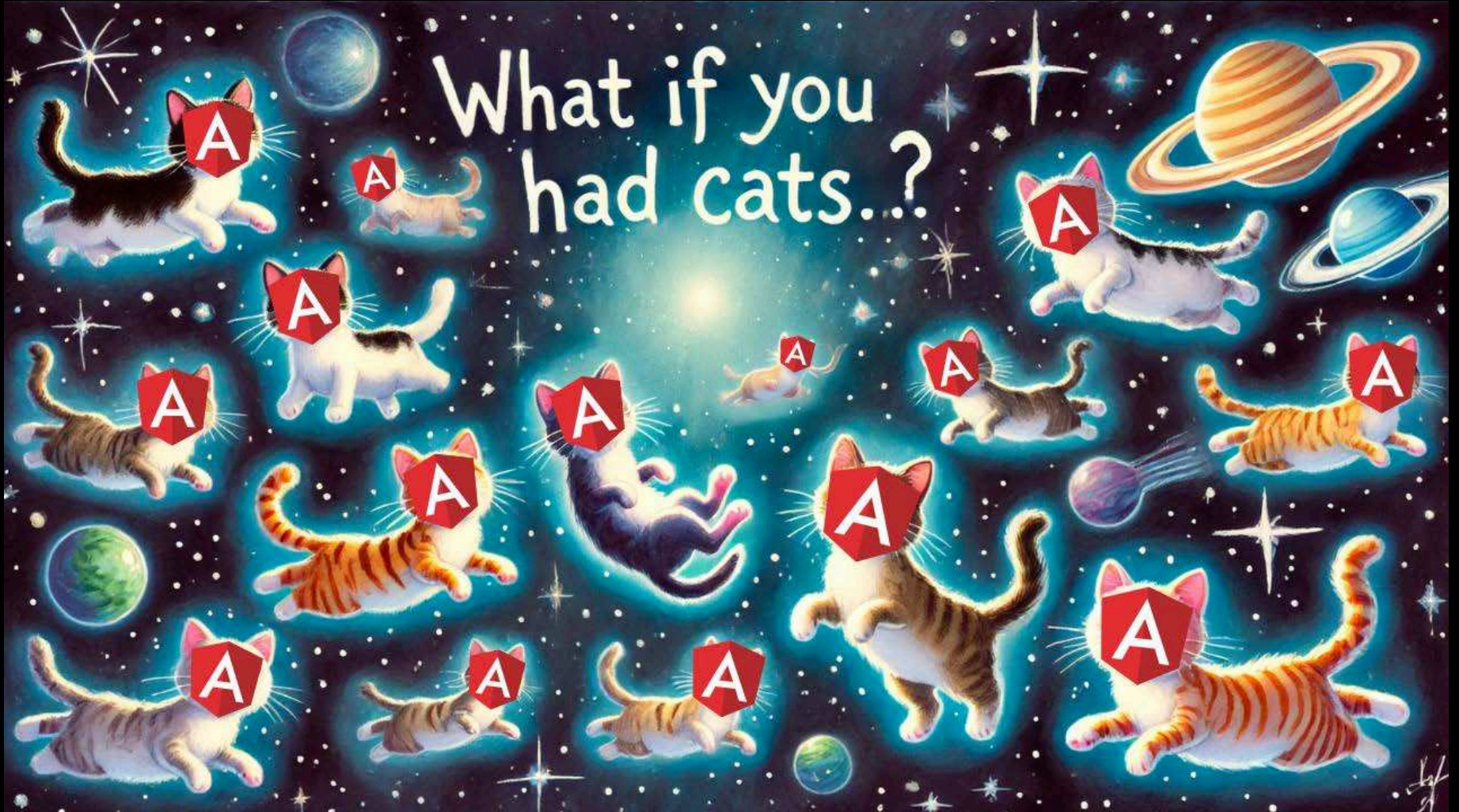


Because it was too late 🙄

Because it was too late 🙄

I already had created the apps, and the generator and templates work when creating new apps

But what about my cats??



I started thinking about the solution

My main questions were:

My main questions were:

- How do geniuses at Angular and NX solve such issues?

My main questions were:

- How do geniuses at Angular and NX solve such issues?
- How can I have a graceful fallback if such a workspace-wide process fails?

My main questions were:

- How do geniuses at Angular and NX solve such issues?
- How can I have a graceful fallback if such a workspace-wide process fails?
- How do I ensure I don't have to think about the same problem again?

First, I created the header component in my NX Angular library

First, I created the header component in my NX Angular library

```

  v codewithahsan
    > .vscode
    > dist
    > e2e
    > node_modules
  v packages
    v ng-cb-ui
      v src
        v lib \ header
          <> header.component.html
          🗝 header.component.scss
          TS header.component.spec.ts
          TS header.component.ts
          TS index.ts
          TS test-setup.ts
          🛡 .eslinttrc.ison

```

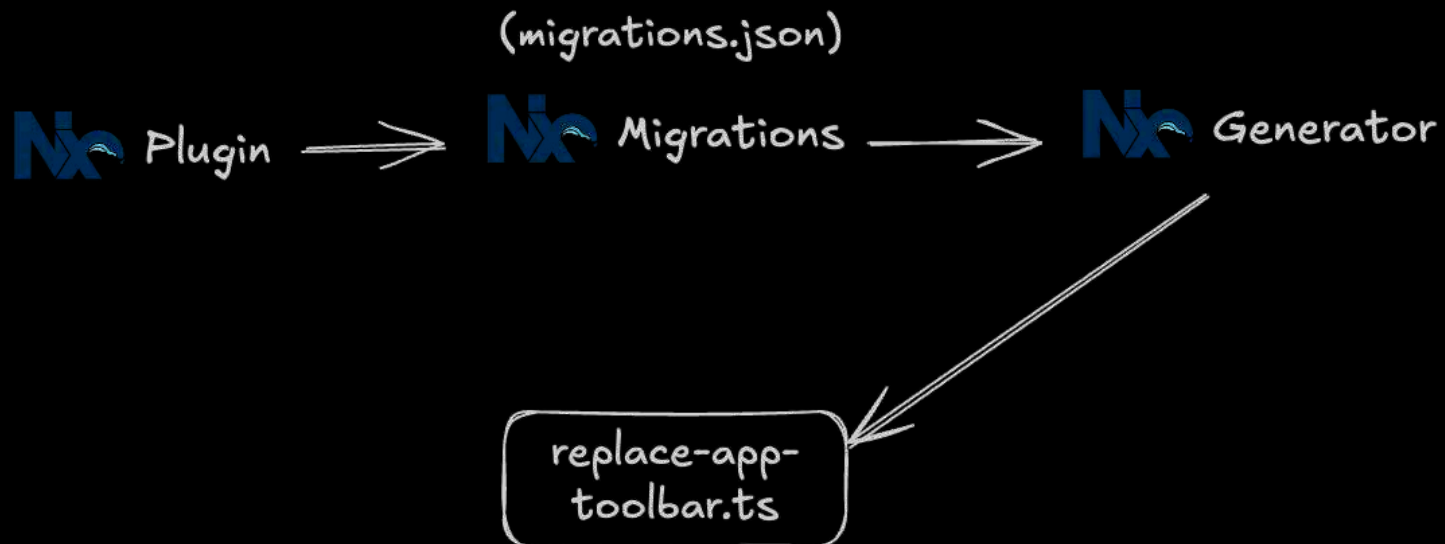
Header Component

Secondly,

Secondly,



Migrations



codewithahsan > packages > ng-cookbook-recipe > {} migrations.json > ...

Muhammad Ahsan Ayaz, 7 months ago | 1 author (Muhammad Ahsan Ayaz)

```
1  {
2    "generators": {
3      "replace-app-toolbar": {
4        "version": "0.0.2",
5        "description": "Updates the toolbar in the apps to use the new header component",
6        "implementation": "./src/migrations/replace-app-toolbar/replace-app-toolbar"
7      }
8    }
9  }
```

The goal was to run the migration
(`replace-app-toolbar`) in both `start` and `final`
workspaces



This is the NX migration script that would replace the toolbars in all the projects

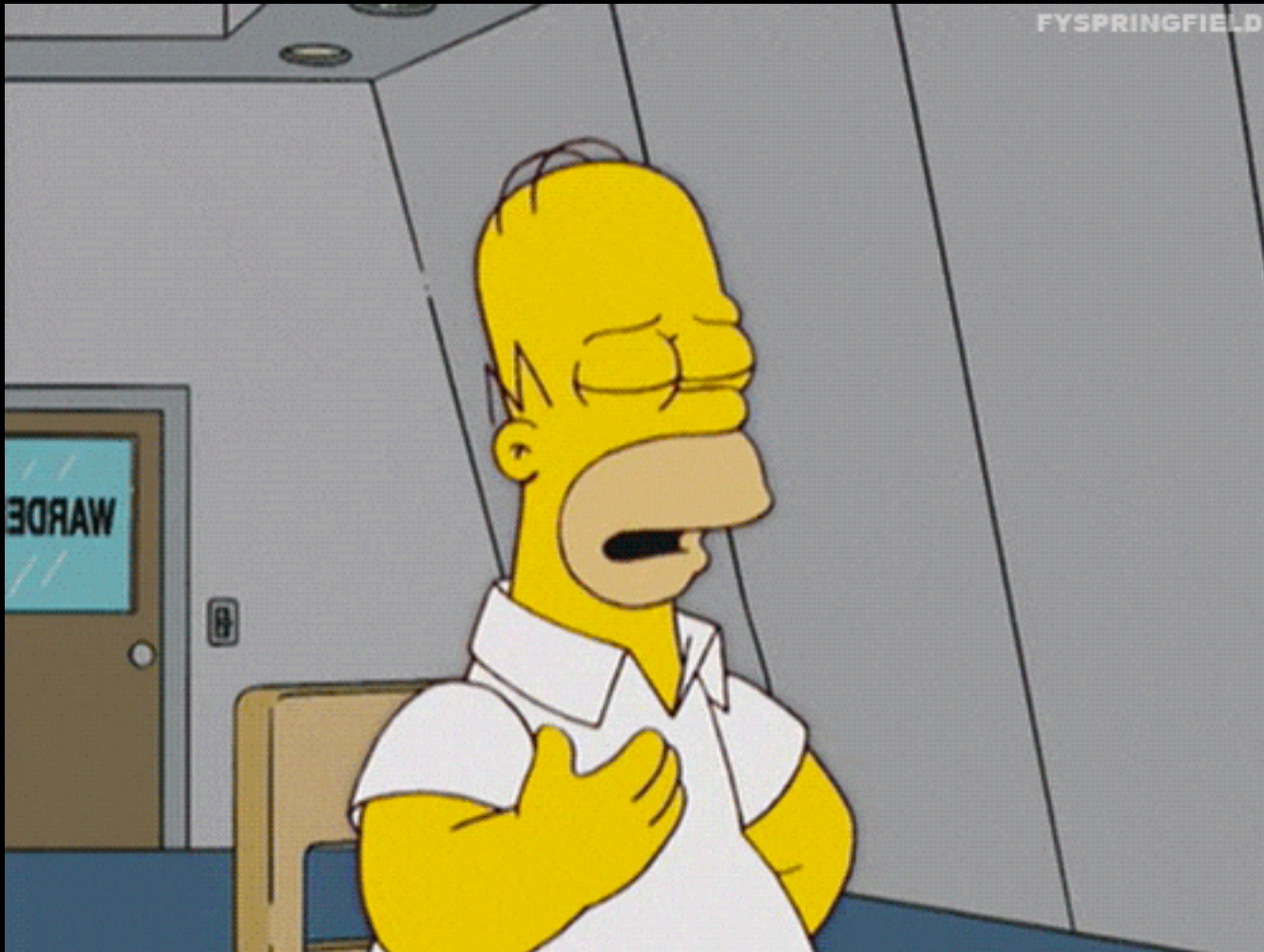
Replace Toolbar Migration

And finally, after running the migrations, I could replace all
the headers

Commit of replacing migrations

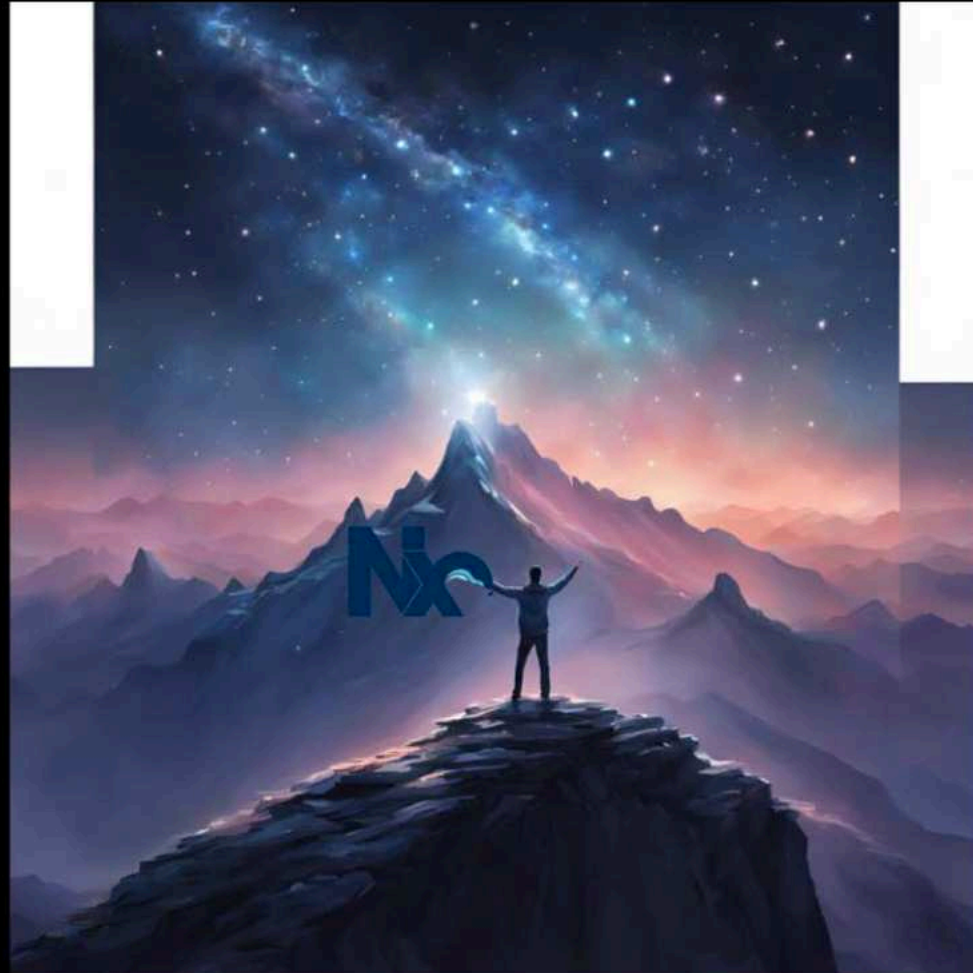
Once I ran the migration with this approach, things were
“maska” as we say in Pakistan.

Once I ran the migration with this approach, things were
“maska” as we say in Pakistan.



And **this** is the final result

I felt like a self-proclaimed NX expert



Summary

Summary

- Cats are ...  pawsome!!

Summary

- Cats are ...  pawsome!!
- Managing 150+ cats(projects) is completely  pawssible!

Summary

- Cats are ... 🐾 pawsome!!
- Managing 150+ cats(projects) is completely 🐾 pawssible!
- No solution is purrfect! We always need to reevaluate and improve our solutions

Summary

- Cats are ... 🐾 pawsome!!
- Managing 150+ cats(projects) is completely 🐾 pawssible!
- No solution is purrfect! We always need to reevaluate and improve our solutions
- It is never too late to learn something new. As soon as the opportunity presents itself, grab it!

Summary

- Cats are ... 🐾 pawsome!!
- Managing 150+ cats(projects) is completely 🐾 pawssible!
- No solution is purrfect! We always need to reevaluate and improve our solutions
- It is never too late to learn something new. As soon as the opportunity presents itself, grab it!
- NX is great for monorepos

Thank you!

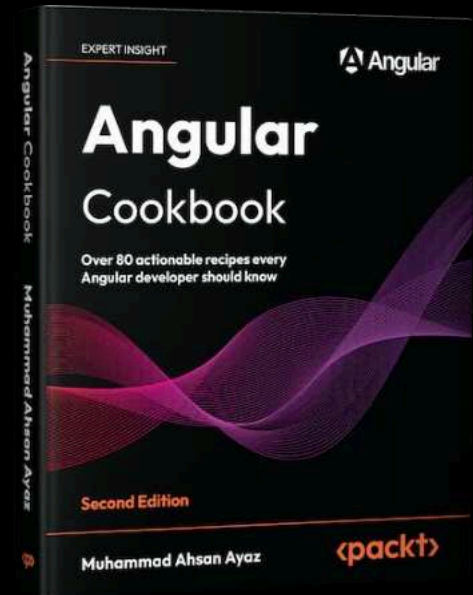


Muhammad Ahsan Ayaz

GDE in Angular

Software Architect at Scania
Group

Co-Founder at VisionWise &
IOMechs



[https://ng-
cookbook.com](https://ng-cookbook.com)



@codewith_ahsan