Front-end architecture in practice 🔧

#whoami

Charly POLY - Senior Software Engineer at algolia

@whereischarly

#whoami

Charly POLY - Senior Software Engineer at algolia

🔗 Writing about Software Eng. on honest.engineering

Writing about TypeScript and GraphQL on Medium

Fun fact

Fun fact


Wes Souza
@WesleydeSouza
HTML is 26 years old.
CSS is 22.
JS is 23.
11:19 pm · 15 Feb 2019 · Twitter Web App
1.6K Retweets
5.4K Likes

Fun fact

- Started to learn JavaScript, 14 years ago, with Prototype

Fun fact

- Started to learn JavaScript, 14 years ago, with Prototype
- JavaScript was a nice way to "mimic" Flash ➡️ animations

JavaScript timeline

JavaScript timeline

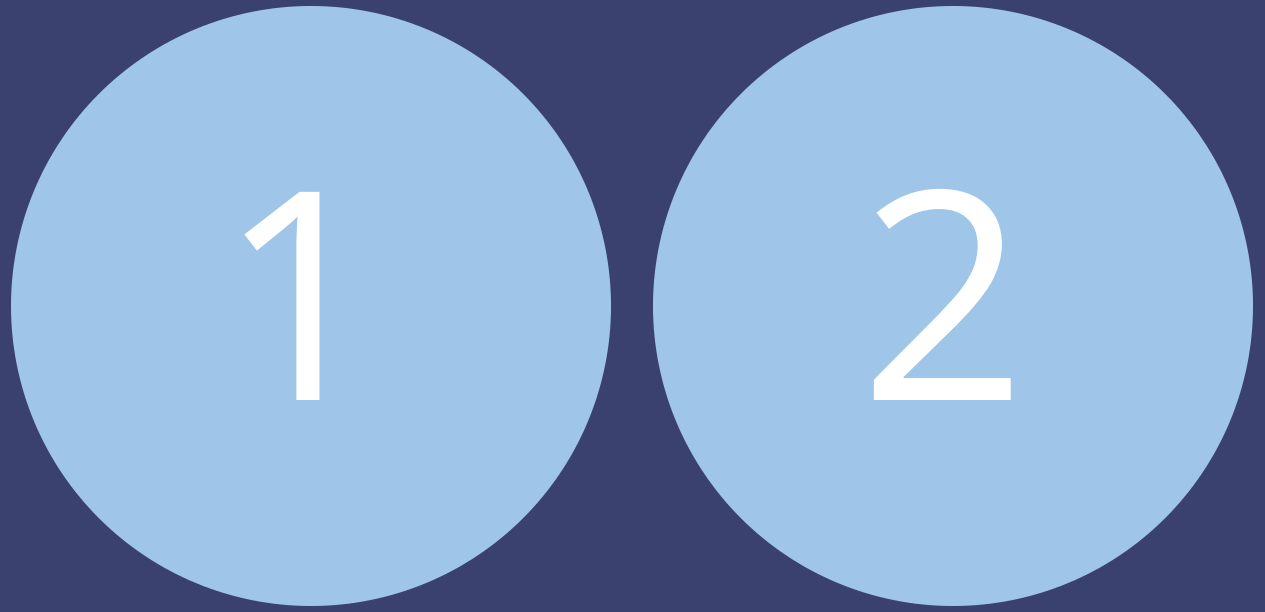
00'

JavaScript timeline

00'

Animations era

JavaScript timeline

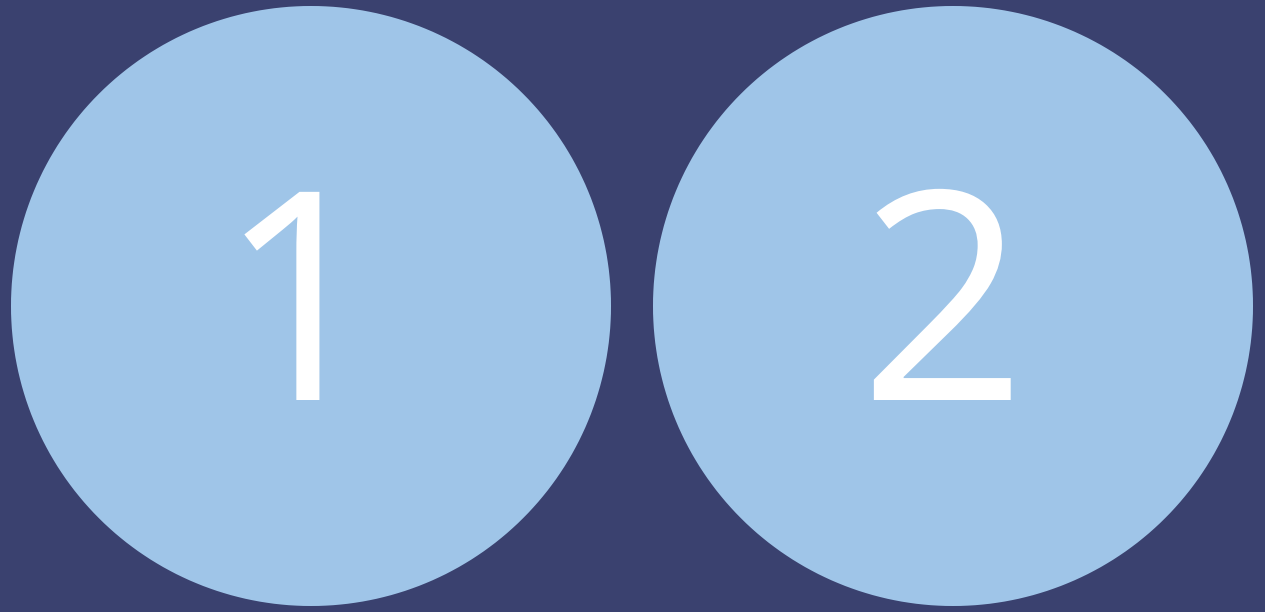


00'

10'

Animations era

JavaScript timeline



00'

Animations era

10'

Interactions era

JavaScript timeline

00'

Animations era

10'

Interactions era

20'

JavaScript timeline

00'

Animations era

10'

Interactions era

20'

SPA era

Front-end evolution

Front-end evolution

1

00'

JQuery /Prototype
Cross-browsers API
Animations
-
-
-

Front-end evolution

1

00'

2

10'

JQuery /Prototype	Backbone
Cross-browsers API	MVP
Animations	Component State management
-	Templating
-	
-	-

Front-end evolution

1 — 00'

2 — 10'

3 — 20'

JQuery /Prototype	Backbone	Angular / React
Cross-browsers API	MVP	MVV, MV*
Animations	Component State management	Separation of Concerns
-	Templating	Application State management
-		
-	-	modules
		DI

Front-end evolution

1

00'

“From animations to apps

3

20'

Modern front-end

Modern front-end

State management

Modern front-end

Routing

State management

Modern front-end

Routing

State management

Crypto

Modern front-end

Routing

State management

Crypto

PWA

Modern front-end

Routing

State management

Crypto

Offline capabilities

PWA

Modern front-end

Routing

State management

Crypto

Offline capabilities

Rendering

PWA

Modern front-end

“ Modern front-end is Software

Modern front-end is Software

Modern front-end is Software

Software Architecture in practice

Modern front-end is Software

Software Architecture in practice

Technical Roadmap

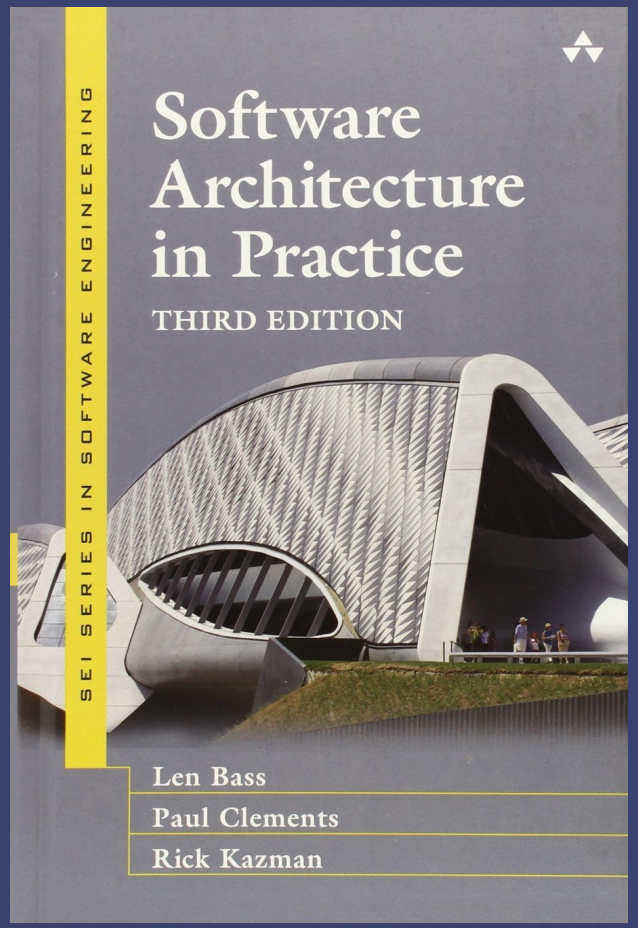
Modern front-end is Software

Software Architecture in practice

Technical Roadmap

Guidelines

Software Architecture in practice



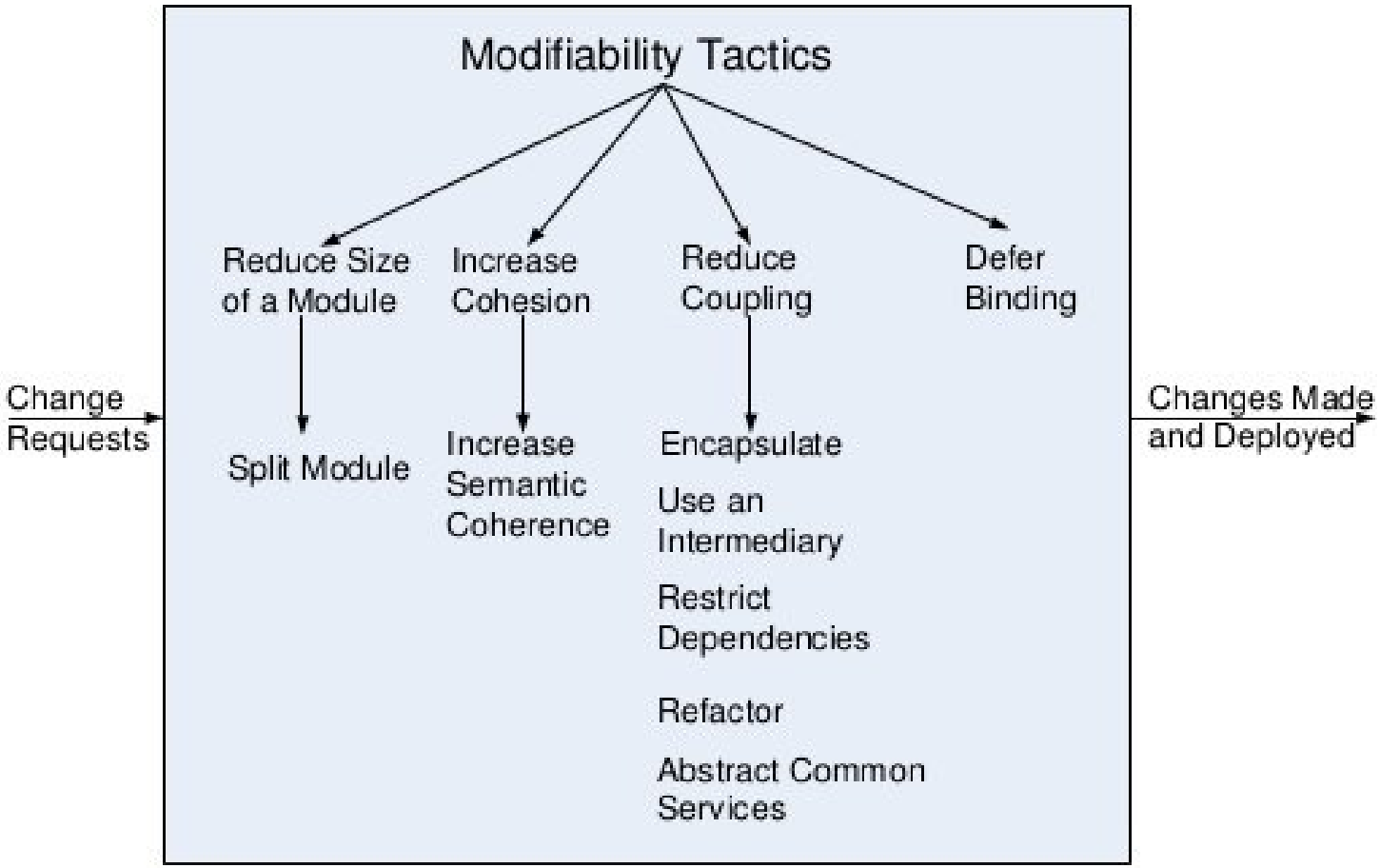
Software Architecture in practice

What is a Quality Attribute?

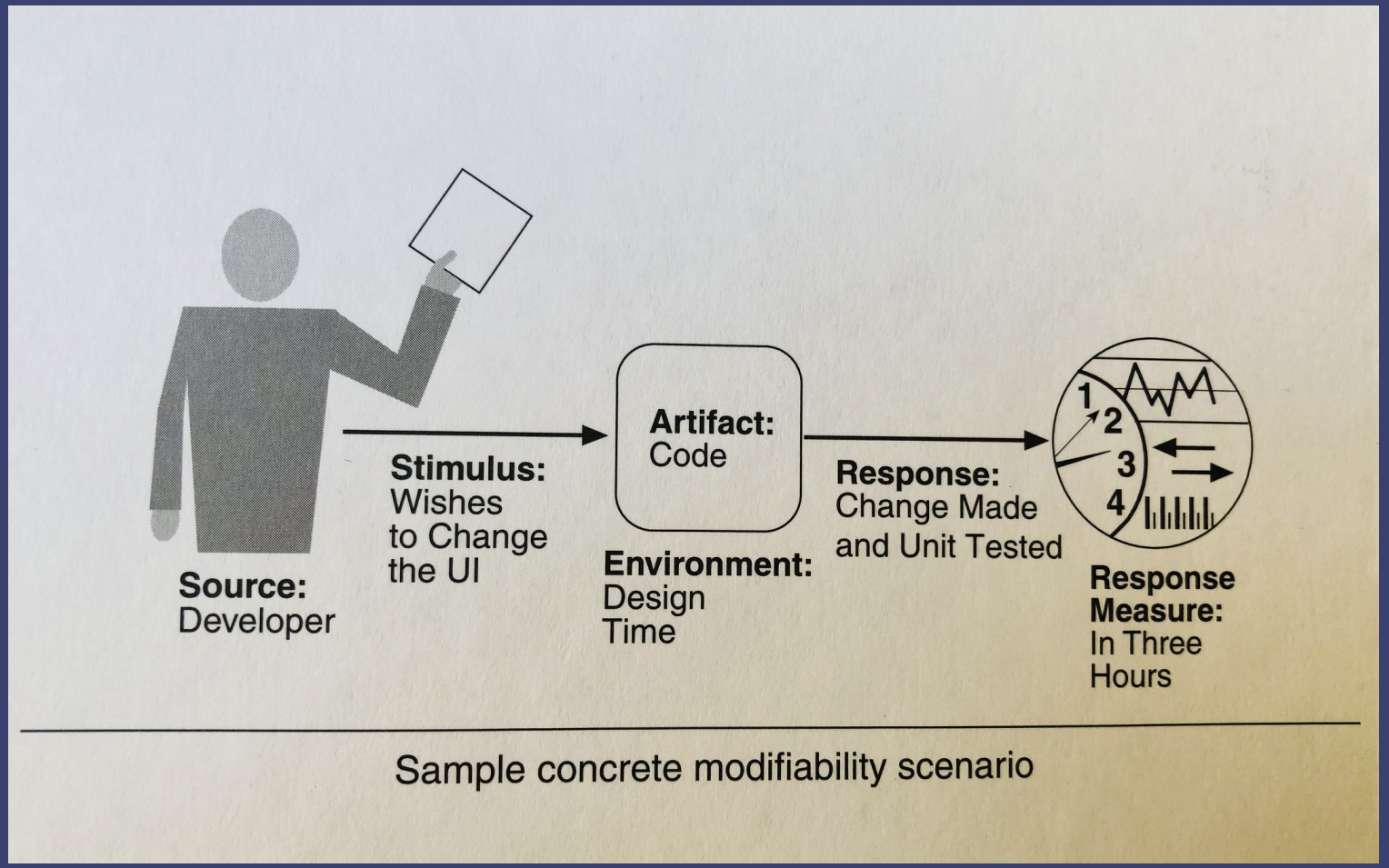
“ A Quality Attribute (QA) is a measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders

Software Architecture in Practice, p.63

Modifiability tactics



Modifiability scenario



Software Architecture in Practice

Quality Attribute sum-up

Quality Attribute = Definition + Tactics + Scenarios

What Quality Attributes for Front-end applications?

What Quality Attributes for Front-end applications?

- **Modifiability**
  Ability to introduce new features, refactor

What Quality Attributes for Front-end applications?

- **Modifiability**
  Ability to introduce new features, refactor

- **Maintainability (Analyzability)**
  Ability to onboard new developers, scale the code

What Quality Attributes for Front-end applications?

- **Modifiability**
  Ability to introduce new features, refactor
- **Maintainability (Analyzability)**
  Ability to onboard new developers, scale the code
- **Performance**
  User perceived performance

What Quality Attributes for Front-end applications?

- **Modifiability**
  Ability to introduce new features, refactor
- **Maintainability (Analyzability)**
  Ability to onboard new developers, scale the code
- **Performance**
  User perceived performance
- **Portability**
  Isomorphism, Browser compatibility, etc ...

What Quality Attributes for Front-end applications?

- **Modifiability**
  Ability to introduce new features, refactor
- **Maintainability (Analyzability)**
  Ability to onboard new developers, scale the code
- **Performance**
  User perceived performance
- **Portability**
  Isomorphism, Browser compatibility, etc ...
- **Security**

Put Architecture in practice

Put Architecture in practice

Start-up stage
(team size, etc..)

AND /
OR

Product
requirements

Put Architecture in practice

Start-up stage (team size, etc..)

AND / OR

Product requirements

Architecture QAs

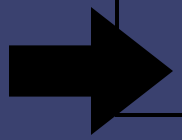
Put Architecture in practice

Start-up stage (team size, etc..)

AND / OR

Product requirements

Architecture QAs



Features updates

Technical roadmap

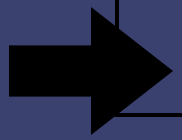
Put Architecture in practice

Start-up stage (team size, etc..)

AND / OR

Product requirements

Architecture QAs



Features updates

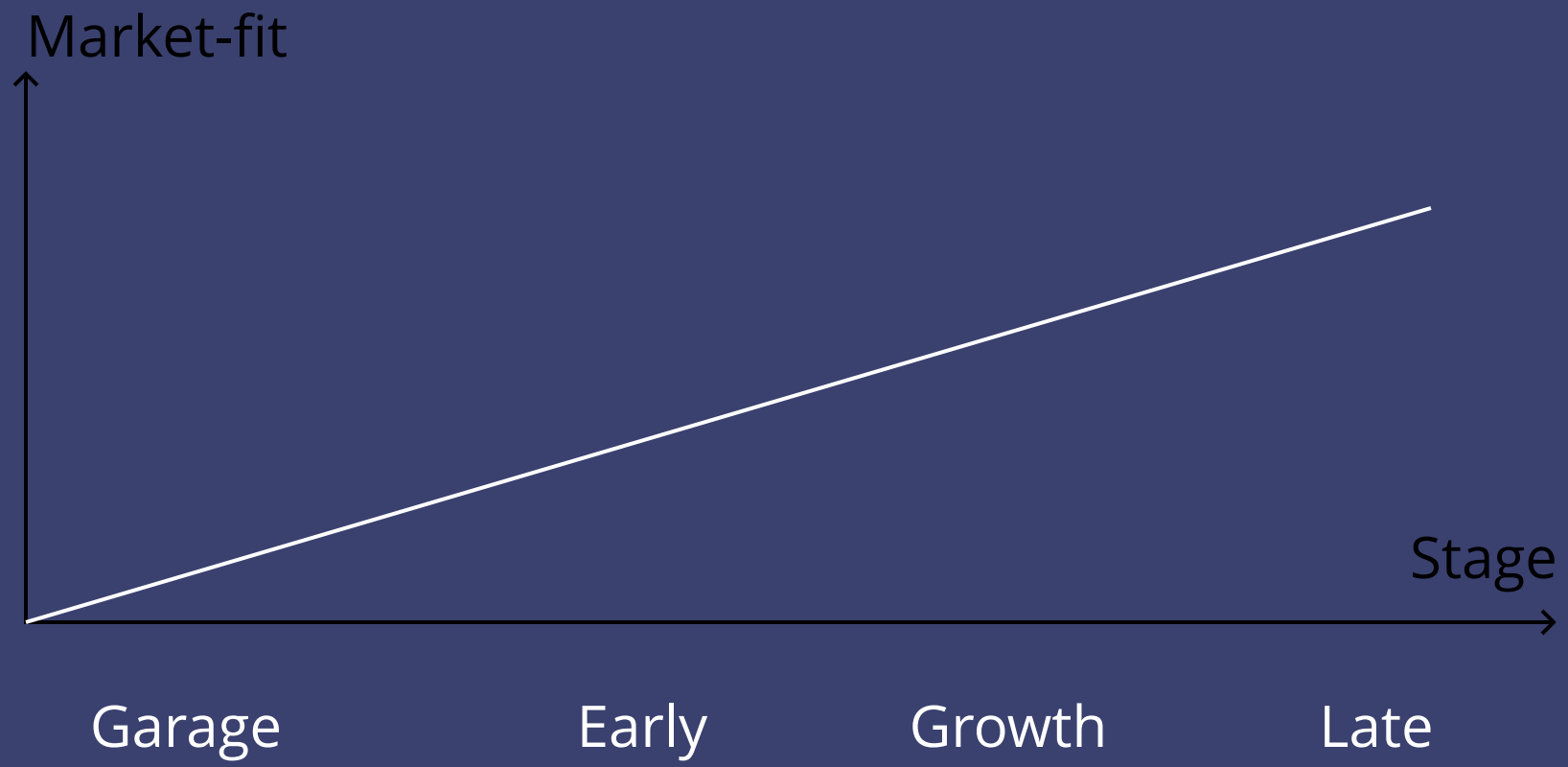
New features

Technical roadmap

Technical guide

How to prioritize Quality Attributes

How to prioritize Quality Attributes: Start-up stage



- **Garage - Growth**
  Modifiability
- **Late**
  Maintainability

How to prioritize Quality Attributes: Product

How to prioritize Quality Attributes: Product

- Security
- Portability
- Maintainability

How to prioritize Quality Attributes: Product

- Security
- Portability
- Maintainability

- Modifiability
- Maintainability
- Performance

Case study: A line front-end architecture

Case study: A line front-end architecture

- **Modifiability**
  Ability to introduce new features, refactor
- **Maintainability**
  Ability to onboard new developers, scale the code
- **Performance**
  User perceived performance

Case study: A line front-end architecture

Build a technical roadmap

Case study: A line front-end architecture

Build a technical roadmap

- TypeScript with types missing everywhere

  ❌ Modifiability, Maintainability ➡️ **type everything!**
- REST API, local custom redux cache

  ❌ Performance ➡️ **GraphQL migration**
- Components with bad naming

  ❌ Maintainability ➡️ **better naming**
- Complex generic class based components

  ❌ Modifiability, Maintainability ➡️ **refactor**
- Complex generic "models" with cache system

  ❌ Modifiability, Maintainability ➡️ **refactor**

Case study: A line front-end architecture

Build technical guidelines

Case study: A line front-end architecture

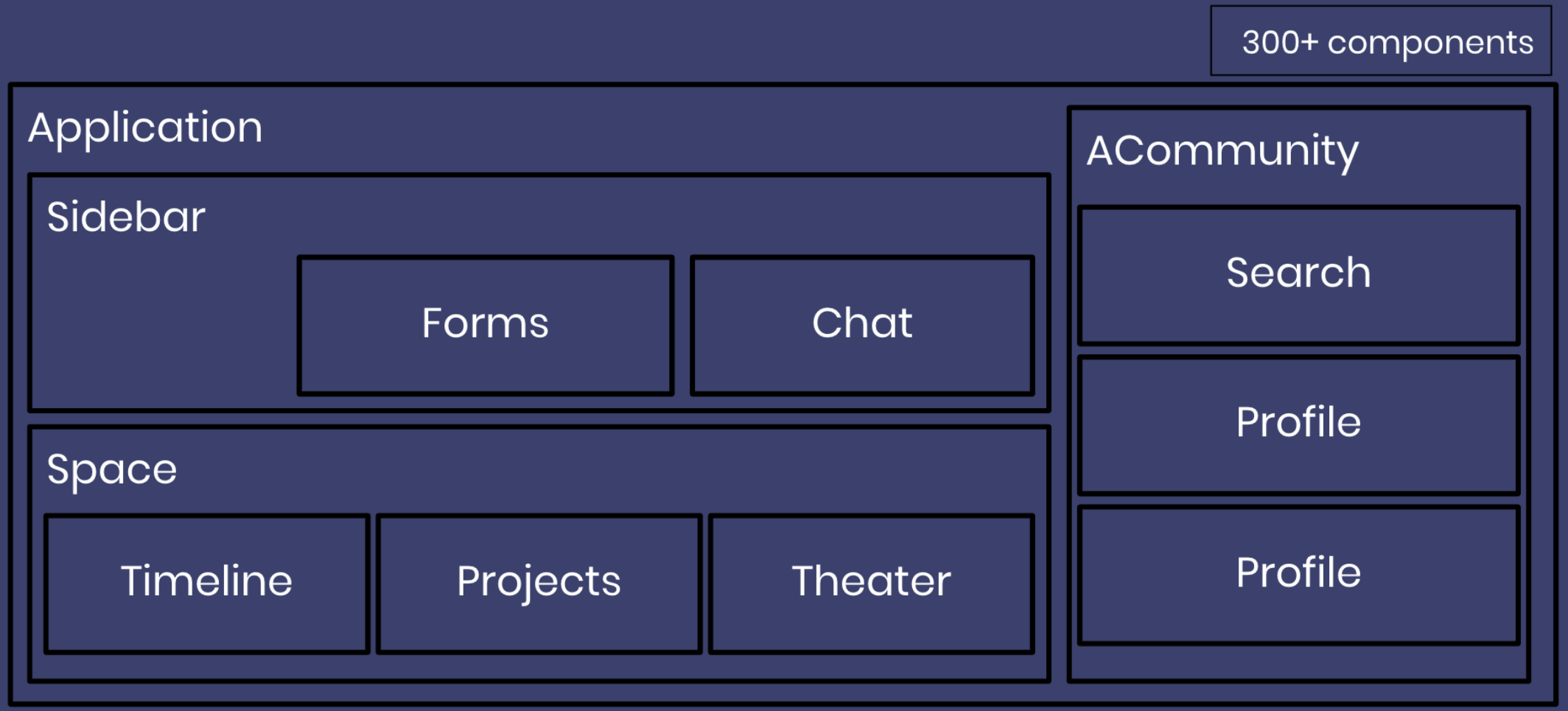
Guidelines for new development

➡️ Enhance **Modifiability**, so :
reduce module size, increase cohesion and reduce coupling

- Components naming convention
- Redux as event-bus to defer binding of components
- Components refactoring to follow SRP principles
- Data HoC in order to prepare GraphQL migration

Case study: A line front-end architecture

Guidelines for new development: Components naming convention



Case study: A line front-end architecture

Guidelines for new development: Components naming convention

Case study: A line front-end architecture

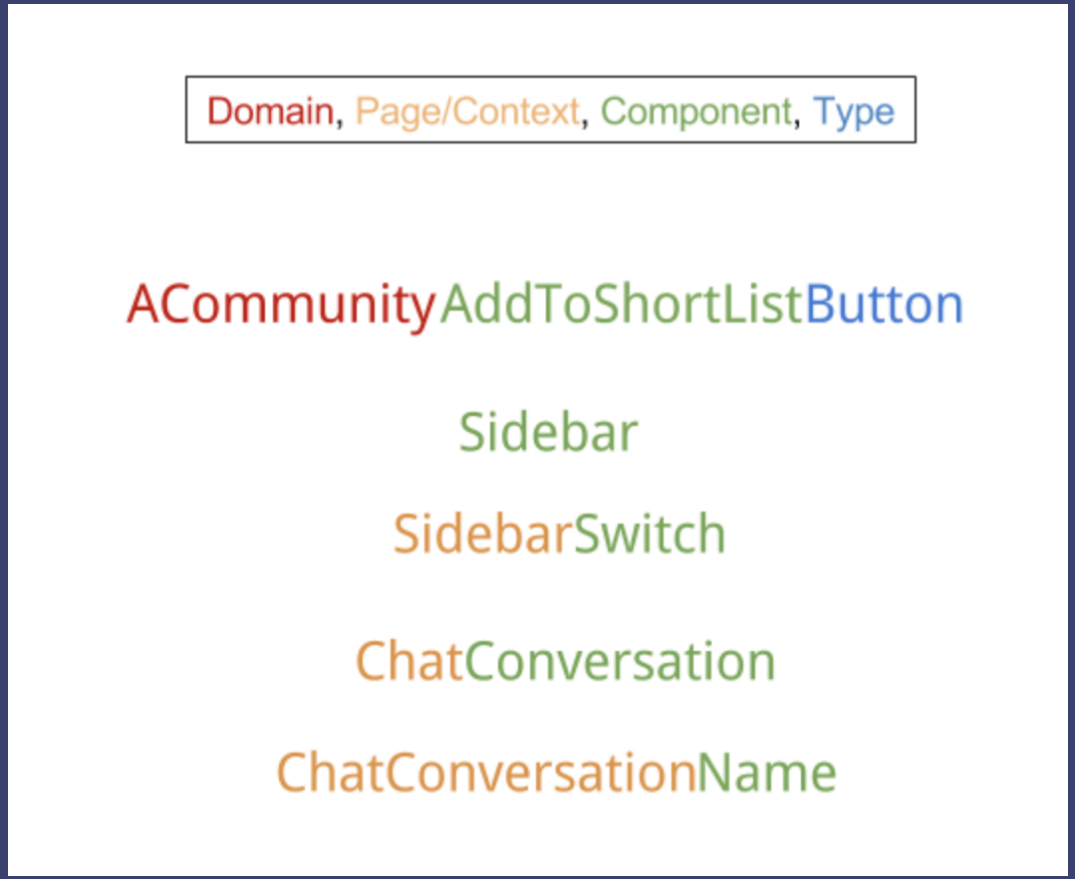
Guidelines for new development: Components naming convention

```
[Domain]|[Page/Context]|ComponentName|[Type]
```

Part surrounded by “[]” are optional.

Case study: A line front-end architecture

Guidelines for new development: Components naming convention

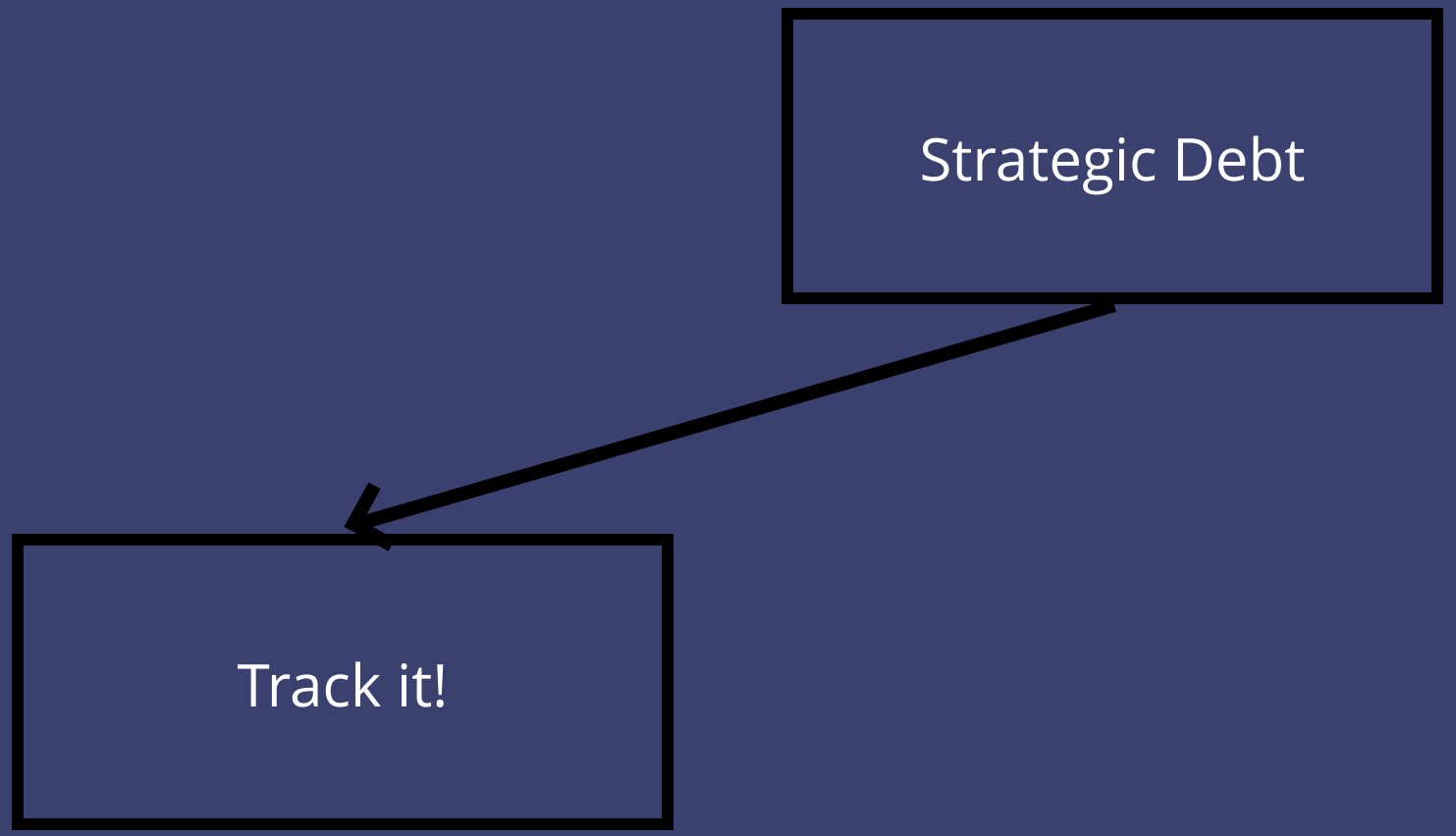


Case study: A line front-end architecture

From redux + REST to GraphQL

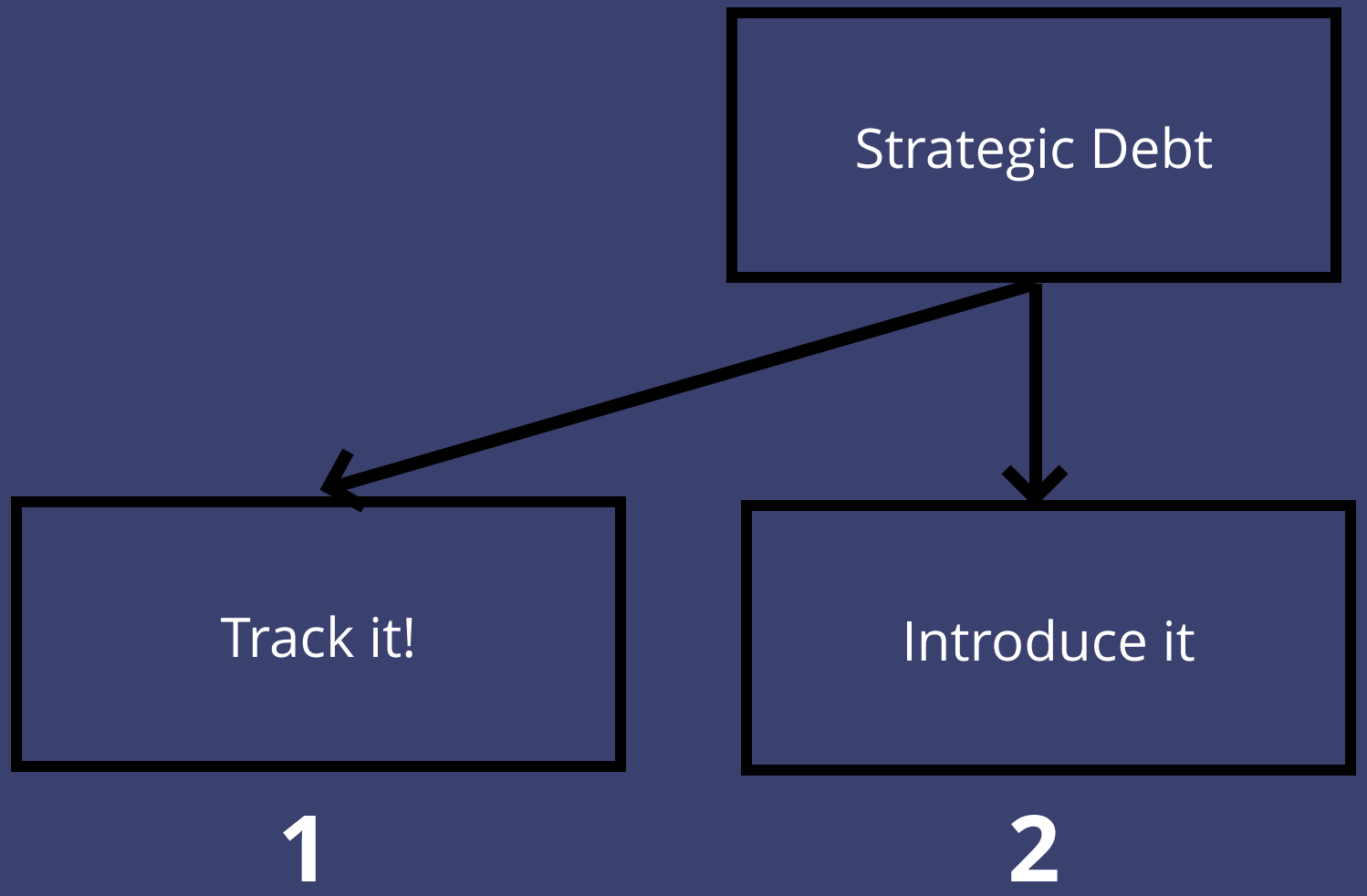
Case study: A line front-end architecture

Case study: A line front-end architecture

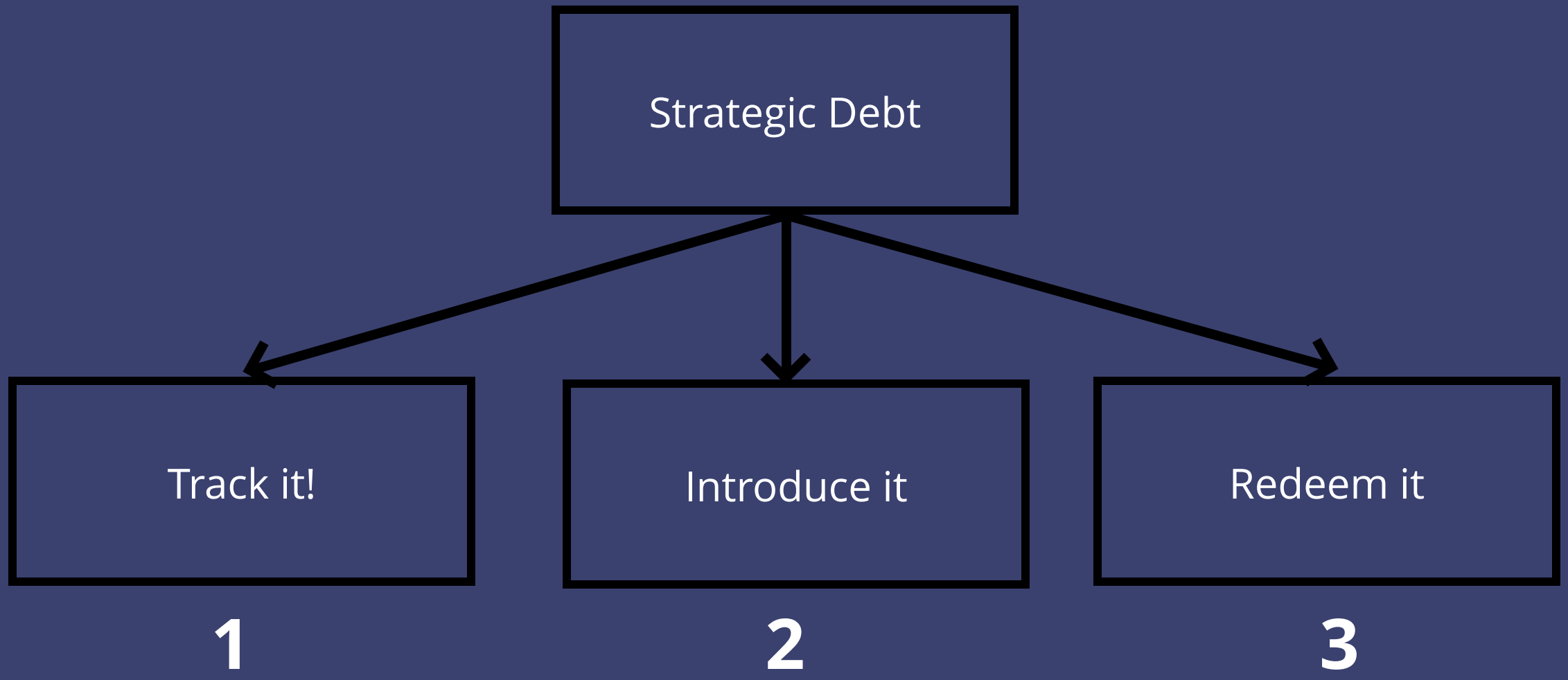


1

Case study: A line front-end architecture



Case study: A line front-end architecture



Case study: A line front-end architecture

Guidelines for new development: REST to GraphQL migration

```
interface Props {
  user: User;
}

export class UserChats extends React.Component<Props, State> {
    componentDidMount() {
        // lot of code
        //      to load data from multiple
        //          REST endpoints
    }

    prepareMyData() { /* ... */ }

    render() {
      const { chats, user } = this.prepareMyData();
        // ...
    }
}
```

Case study: A line front-end architecture

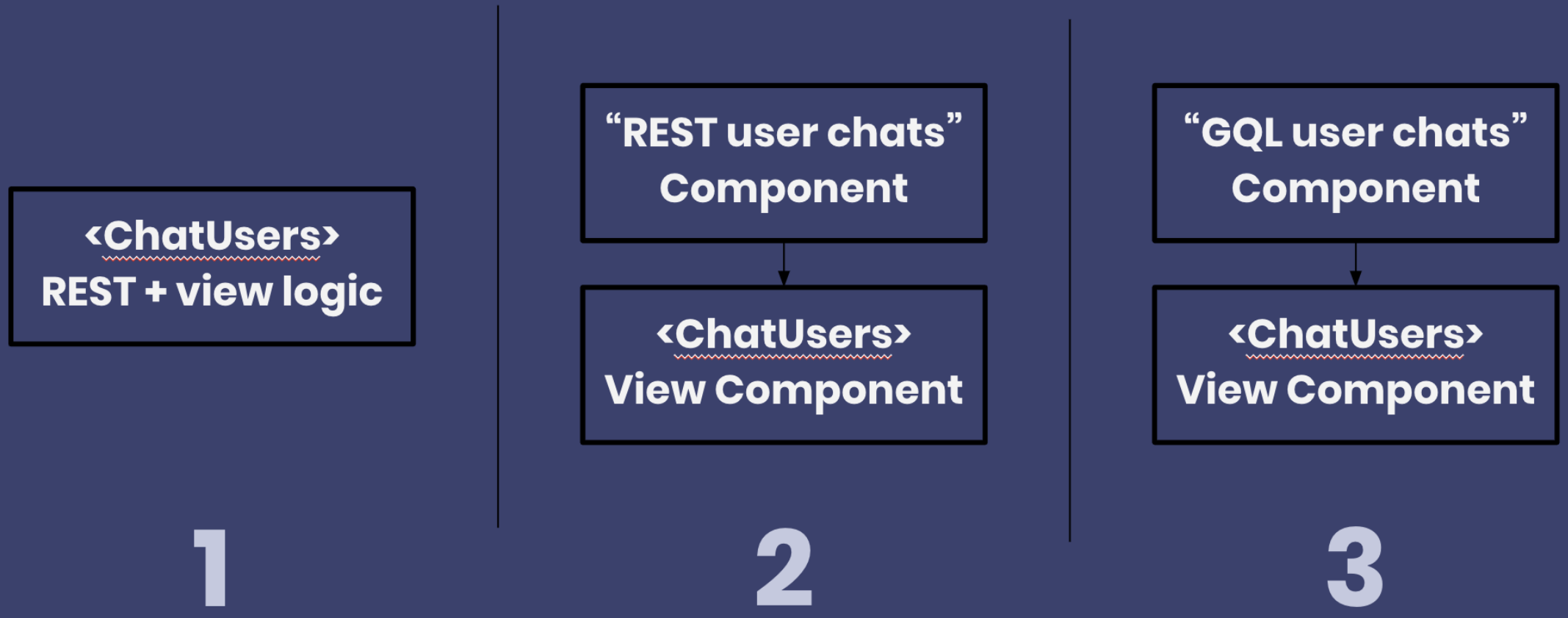
Guidelines for new development: REST to GraphQL migration

```
interface Props {
  user: User;
  chats: Chat[];
}

export class UserChatsView extends React.Component<Props, State> {
    render() {
        // ...
    }
}
// ...
export const UserChats = loadChatsForUser(UserChatsView);
```

Case study: A line front-end architecture

Guidelines for new development: REST to GraphQL migration



Conclusion

Conclusion

- Think architecture, not code

Conclusion

- Think architecture, not code
- Build your roadmap and guidelines by using Quality Attributes

Conclusion

- Think architecture, not code
- Build your roadmap and guidelines by using Quality Attributes
- Architecture good points:
  - clear structure, easier to document
    - enhance team work, onboarding
  - early prediction of system's qualities
  - can be re-usable

Thanks for listening!

See you at React Europe 🎙️ 🇪🇺

🔗 honest.engineering

 @whereischarly

 /wittydeveloper