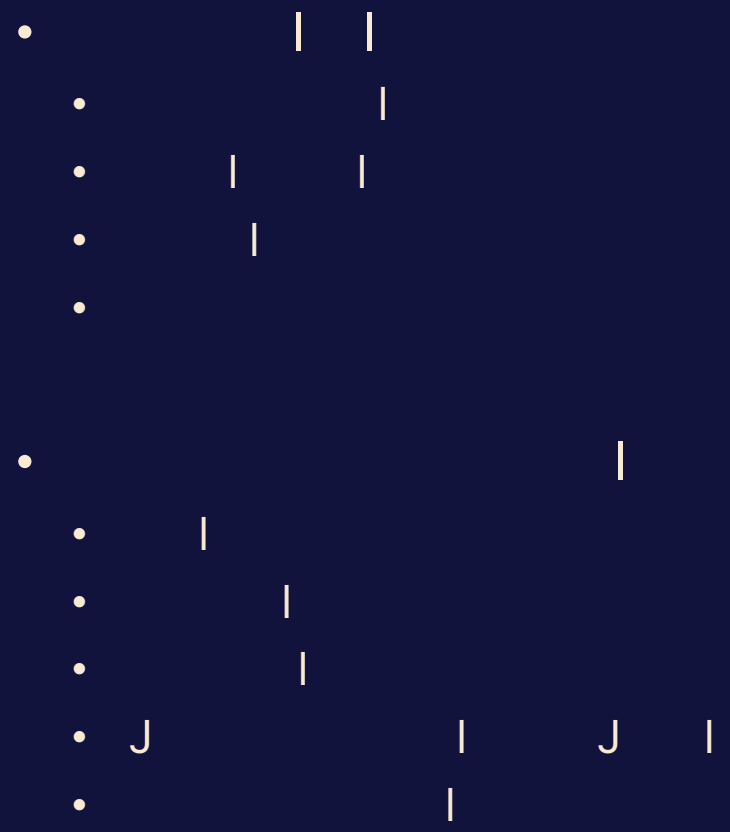
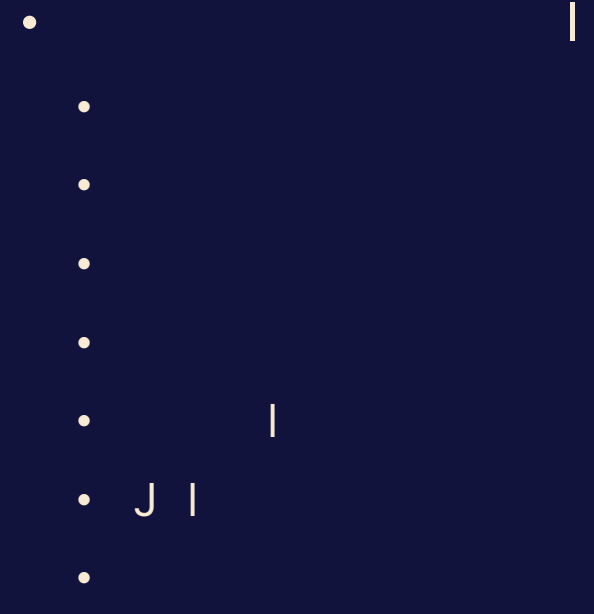
PRACTICAL REACT WITH TYPESCRIPT

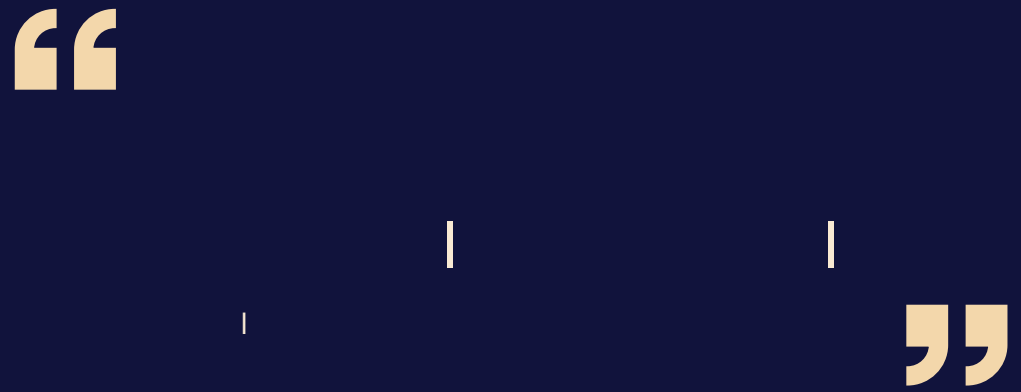
Agenda





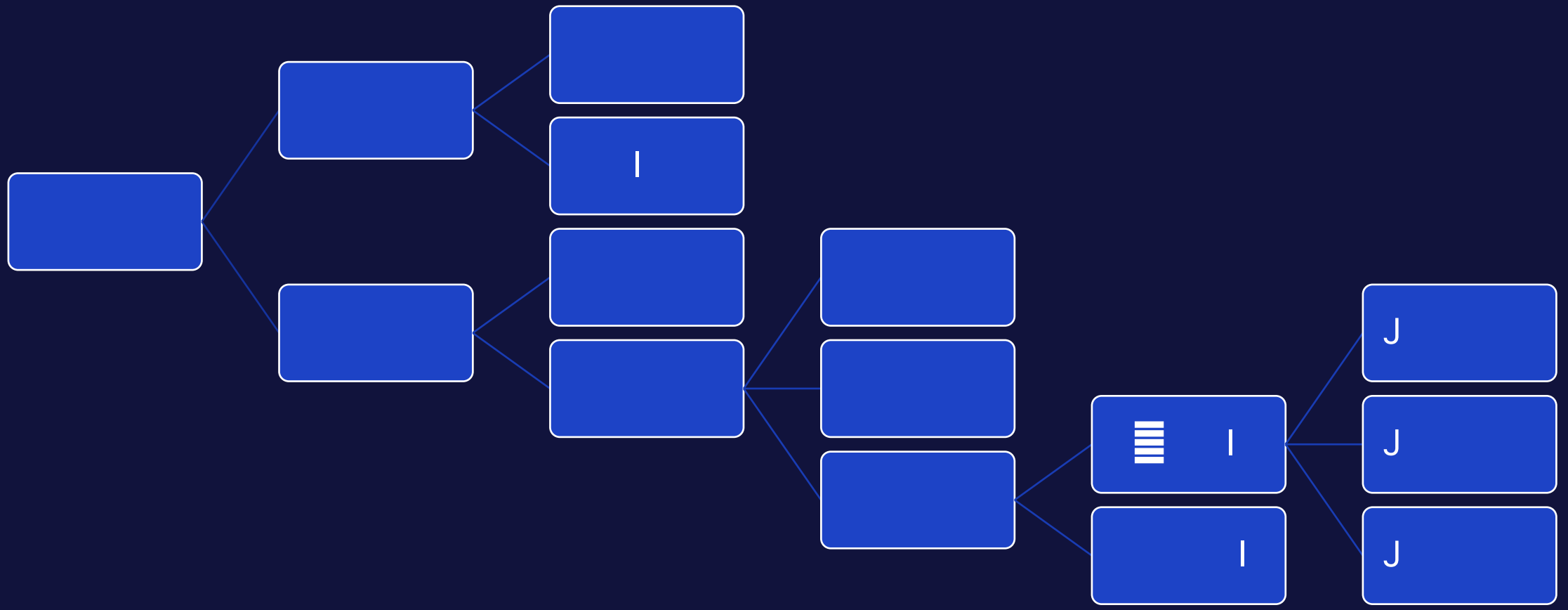
bouvet

React basics



bouvet

Anatomy of React



bouvet

TextField

-
-
-

bouvet

<> BooleanField

- |
  |
- | J
  |

bouvet

Anatomy of a component

```tsx
export interface TextFieldProps {
  label: string
}
```

props

```tsx
export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

component

Anatomy of a component

```tsx
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

bouvet

Anatomy of a component

```
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

use* | hooks
|

bouvet

Anatomy of a component

```tsx
export interface TextFieldProps {
  label: string
}

ex                                                    dProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

useState

Anatomy of a component

```
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
                        |       |
                         |  jsx      |
  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

bouvet

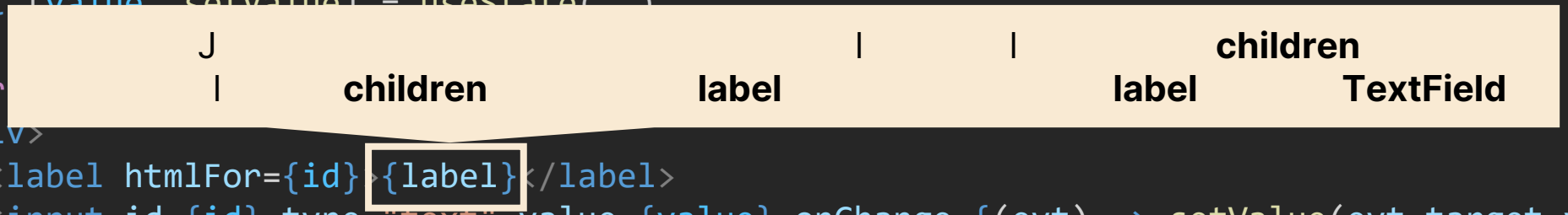
Anatomy of a component

```tsx
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")
  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

htmlFor label id

Anatomy of a component

```tsx
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```



bouvet

Anatomy of a component

useState

destructure

```tsx
export interface TextFieldProps {
  label: string
}

exp
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

Anatomy of a component

```tsx
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

value

input

onChange

function

bouvet

Anatomy of a component

- **Component:**

- **Props**
- **Hooks**
- **State**
- **Children**
- **JSX**
- **{}**

JSX

bouvet

Anatomy of an event

```tsx
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

input*

0 | | | **input** | **onChange** J J | **input**

Anatomy of an event

```tsx
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

event handler

setter

Anatomy of an event

```tsx
export interface TextFieldProps {
  label: string
}
```

State change | re-render updated data

```tsx
export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

bouvet

Anatomy of an event

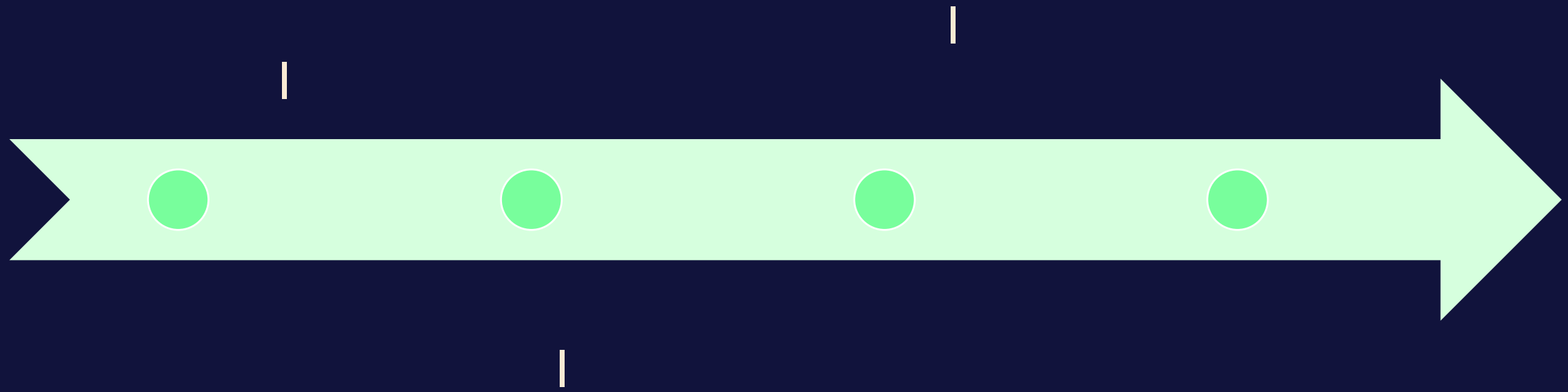
```tsx
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

value

updating the UI

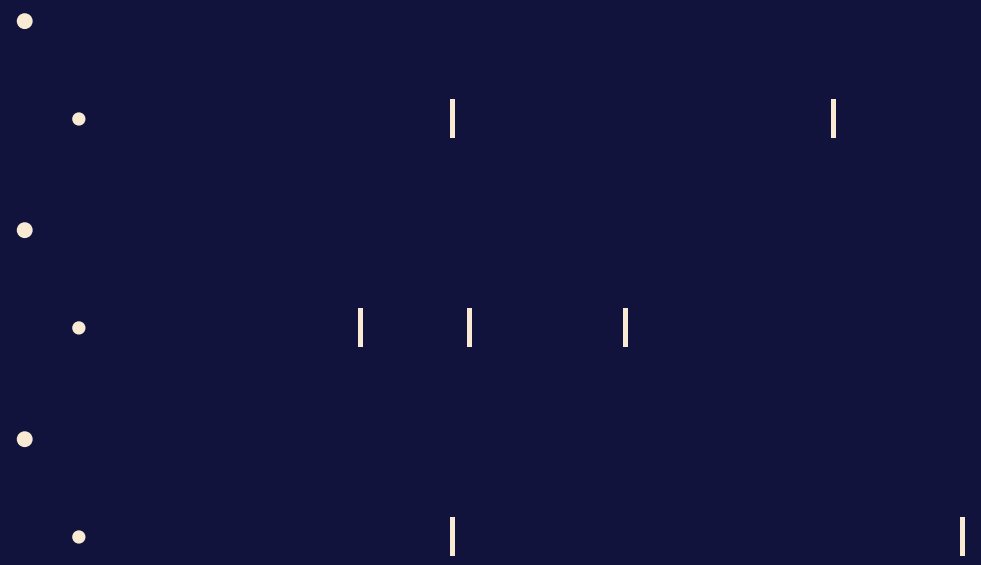
bouvet

Anatomy of an event



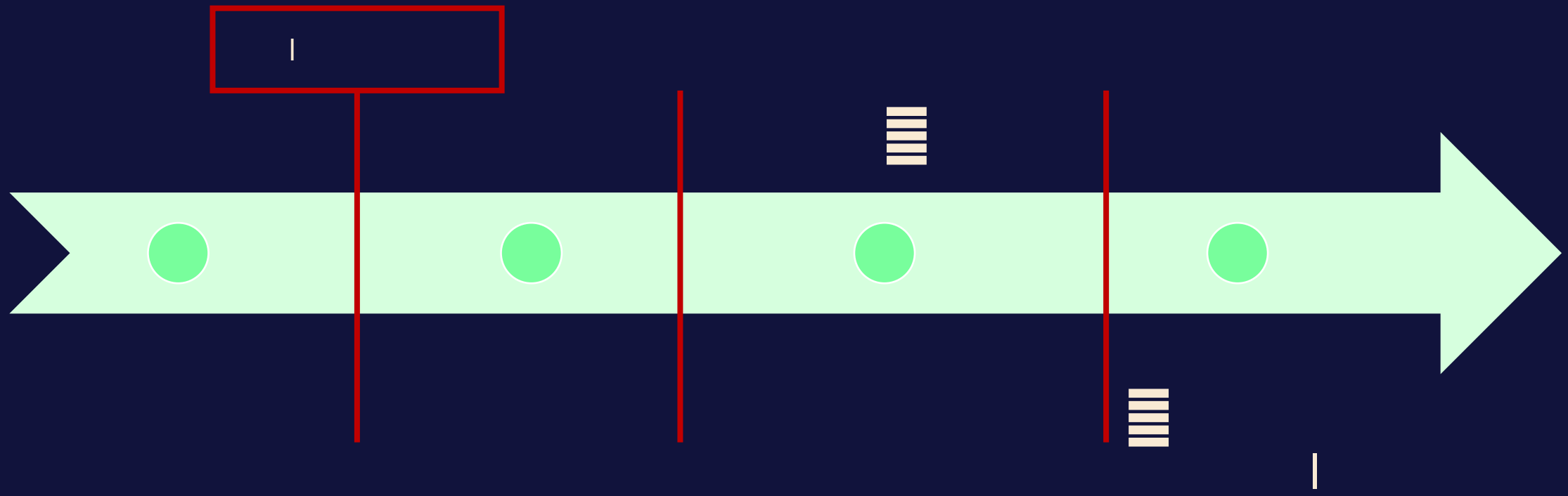
bouvet

Component lifecycle



Component lifecycle

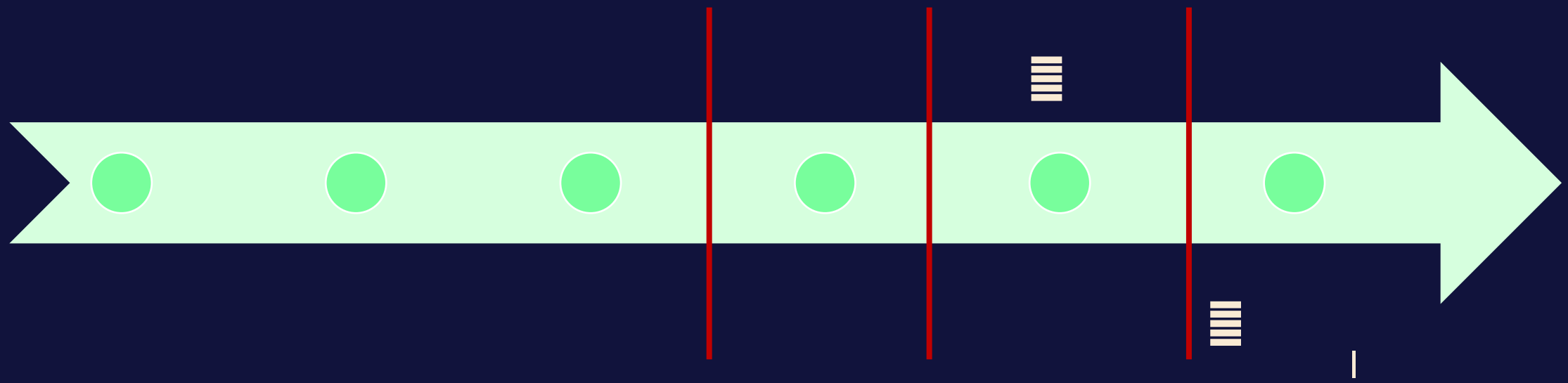
TextField



bouvet

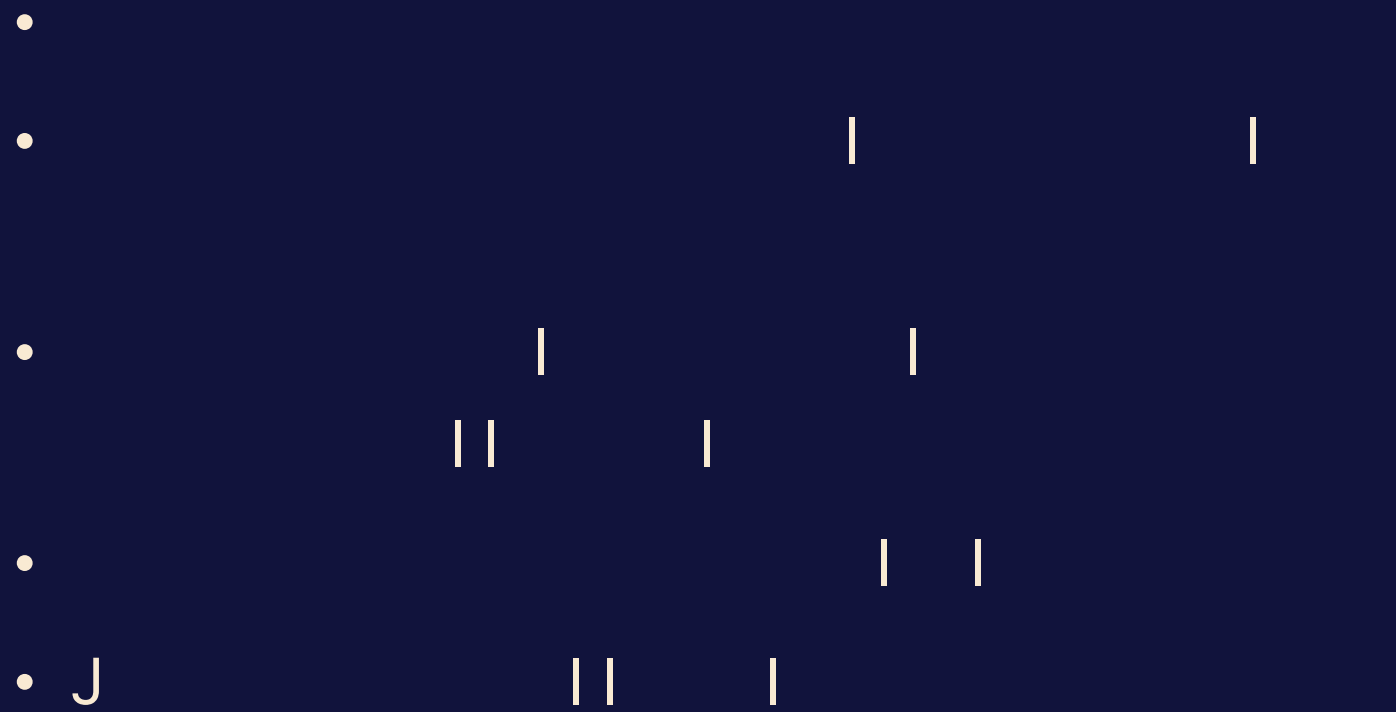
Component lifecycle

TextField

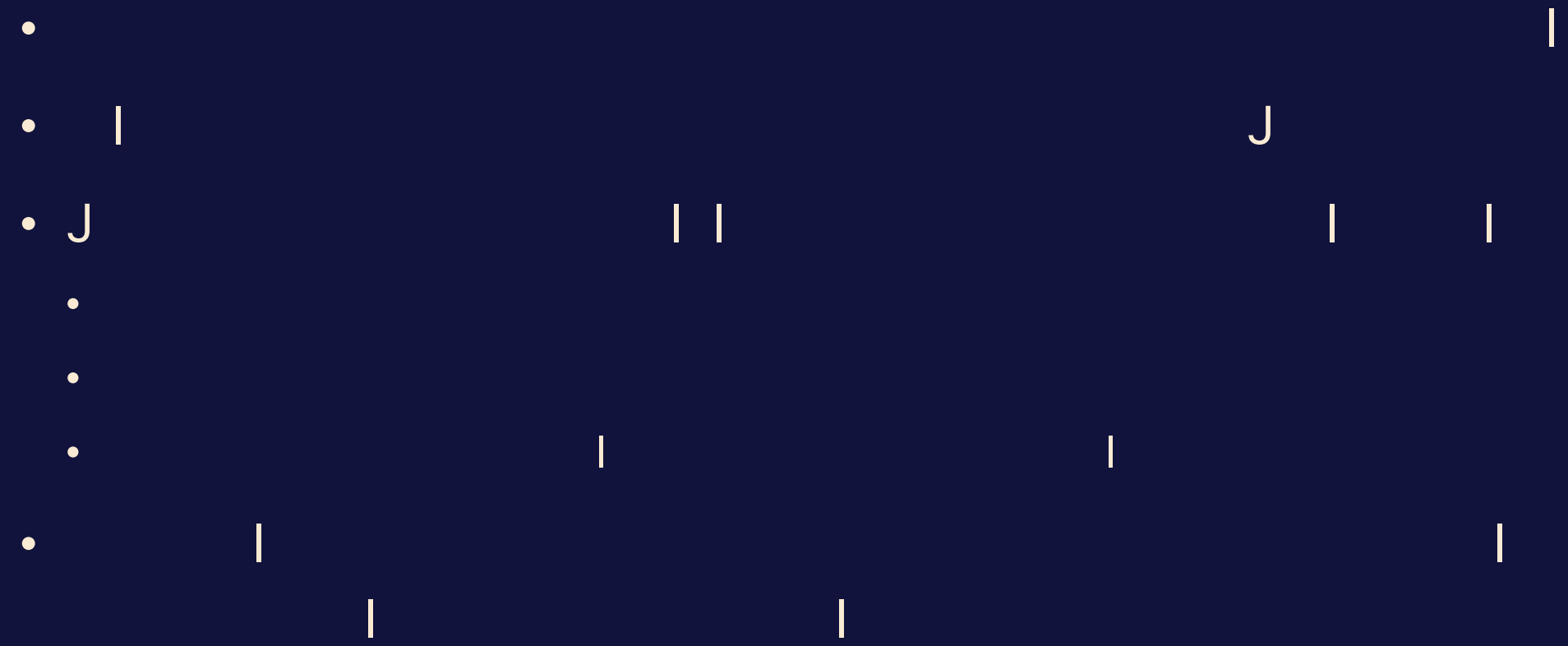


bouvet

ClickUntil



<> NumericField



bouvet

<> ClickUntilForm

bouvet

Styling

-
  -
  -
  -

bouvet

Style TextField

- I I I J

bouvet

<> Style components

-
-
-

bouvet

EoD 1

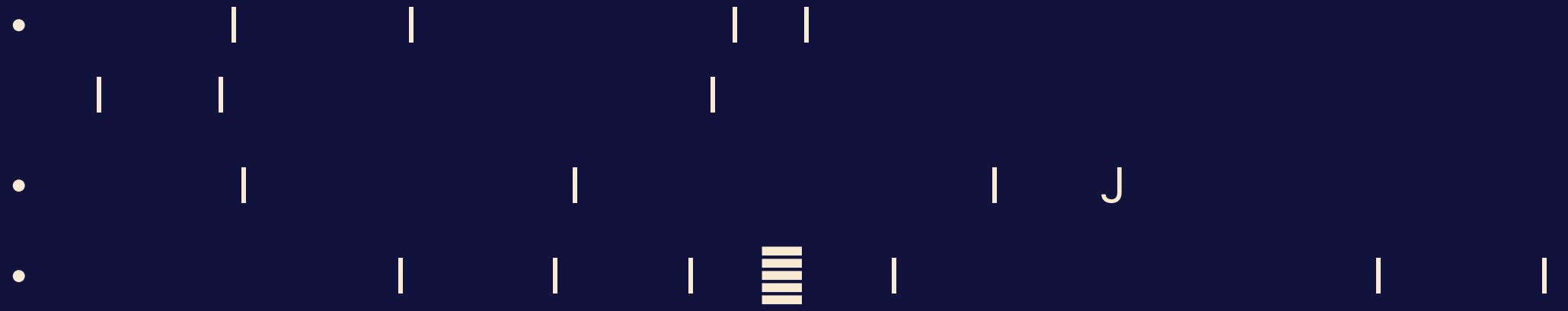
bouvet

Immutability

- 
-

bouvet

UserDetails



UserDetails2

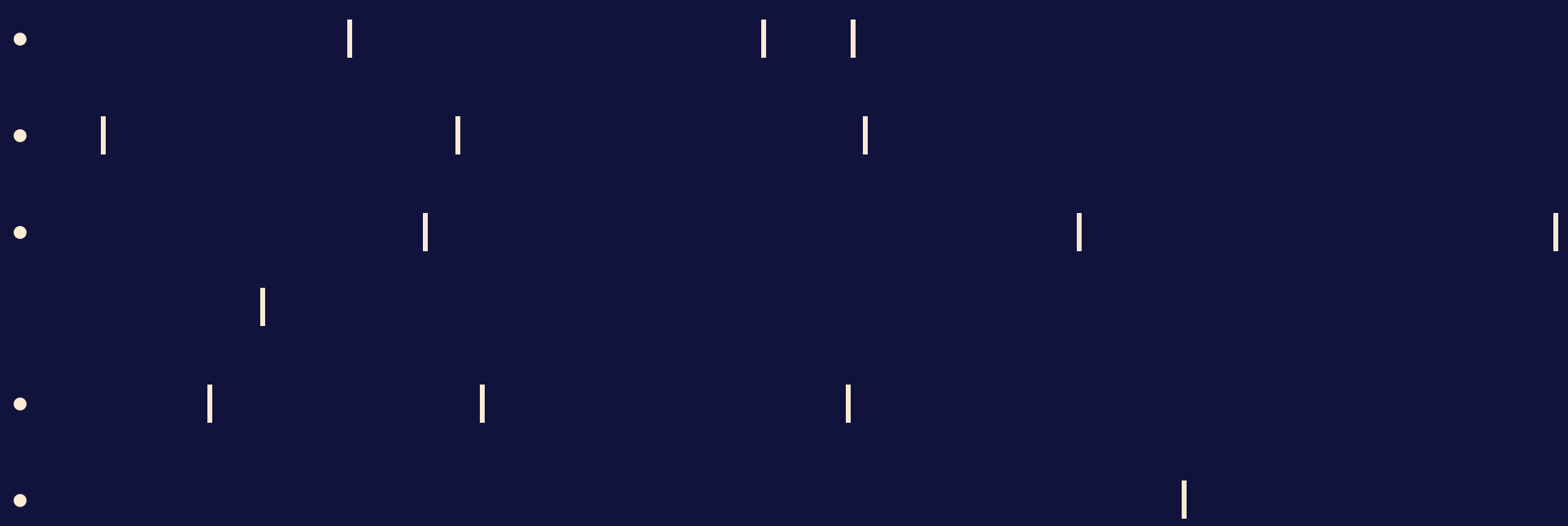
bouvet

Loops

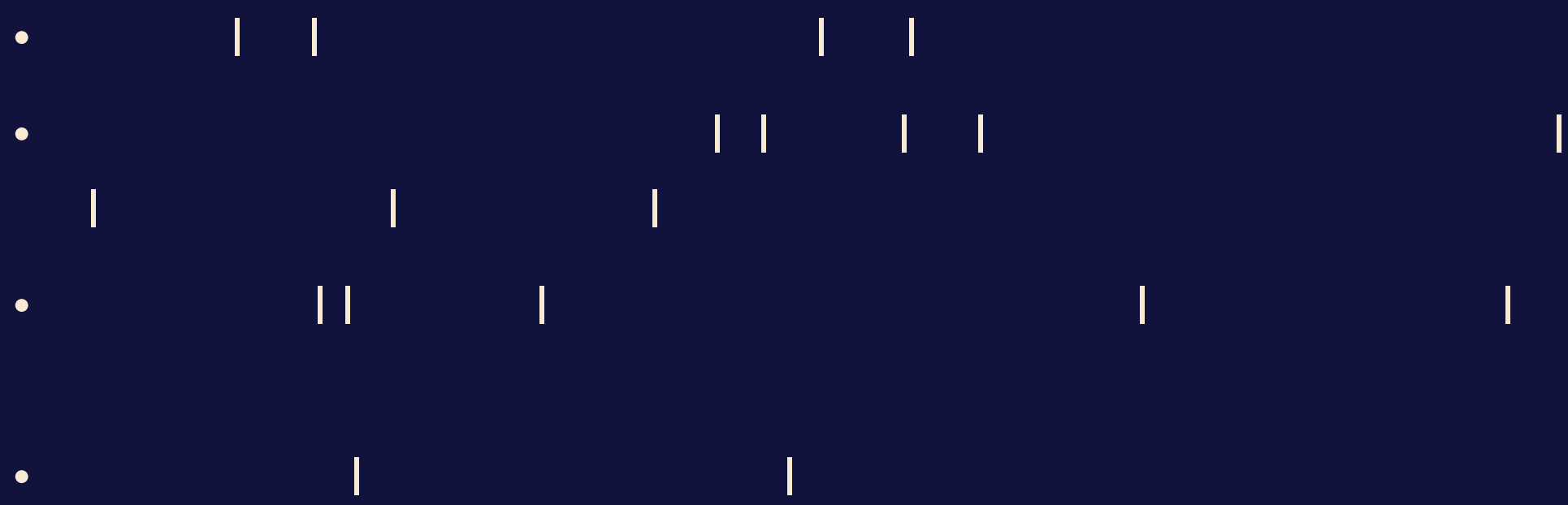
-
-

bouvet

ListData component



<> UsersTable



bouvet

Hooks

-
-
  -

bouvet

Most common react hooks

- **useId:**
- **useState:**
- **useMemo:**
- **useEffect:**
- **useRef:** **does not**

bouvet

useRandom

bouvet

<> useWindowTitle

bouvet

<> useLocalStorage

bouvet

TypeScript Generics

bouvet

TypeScript Generics

```ts
interface ContainerFor<TType> {
  id: string
  value: TType
}
```

```ts
const stringContainer: ContainerFor<string> = {
  id: "123",
  value: "foo"
}

const booleanContainer: ContainerFor<boolean> = {
  id: "123",
  value: true
}

interface Name {
  firstName: string
  lastName: string
}
const objectContainer: ContainerFor<Name> = {
  id: "123123",
  value: {
    firstName: "Test",
    lastName: "Testington"
  }
}
```

TypeScript Generics

```ts
interface ContainerFor<TType extends string | boolean> {
  id: string
  value: TType
}
```

```ts
const stringContainer: ContainerFor<string> = {
  id: "123",
  value: "foo"
}

const booleanContainer: ContainerFor<boolean> = {
  id: "123",
  value: true
}
```

TypeScript Generics

```ts
interface GenericInterface<T> {
  id: string
  value: T
}

type GenericType<T> = {
  id: string
  value: T
}

const genericFn = <T, T2>(firstArg: T, secondArg: T2) => {
  //...
}

function genericFn2<T, TReturn>(firstArg: T): TReturn {
  //...
}
```

bouvet

TypeScript Generics with React

```tsx
export interface GenericComponentProps<TType> {
  label: string
  value: TType
}

export const GenericComponent = <TType,>(props: GenericComponentProps<TType>) => {
  // ..
}
```

bouvet

<> ChoiceField

bouvet

EoD 2

bouvet

Organize the workspace

-
-
-

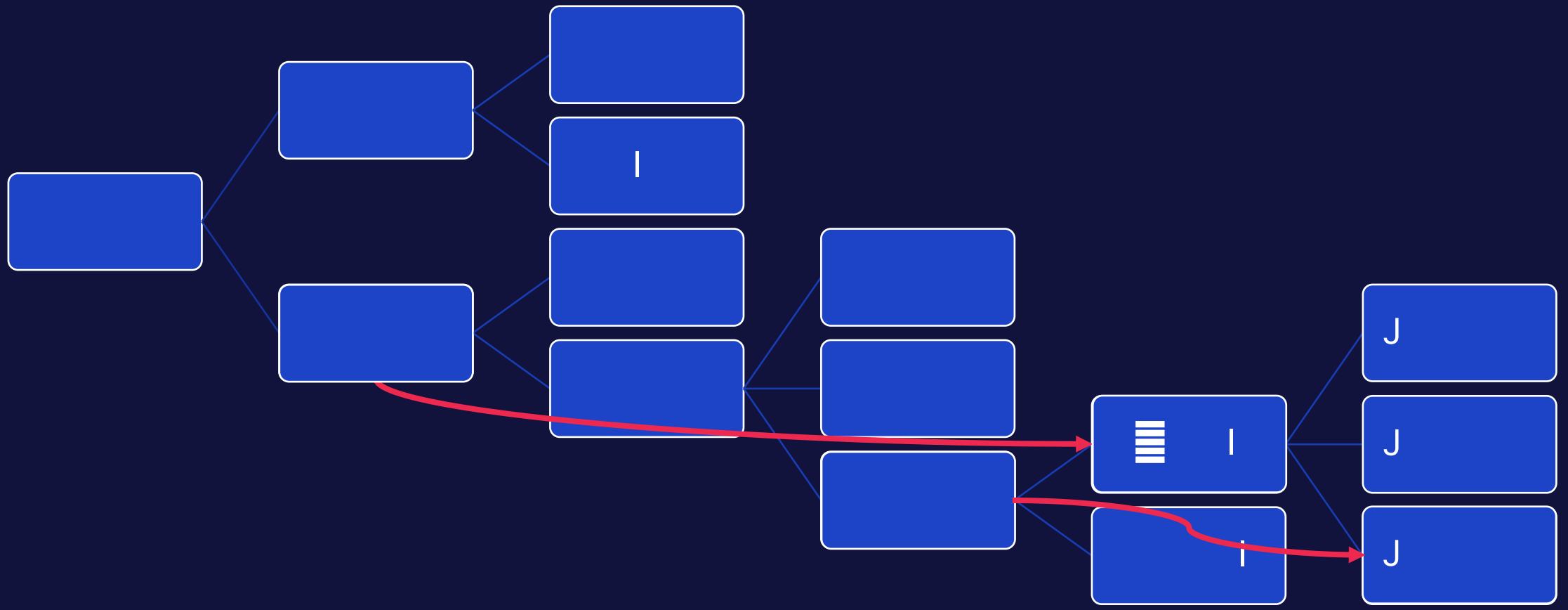
bouvet

Contexts

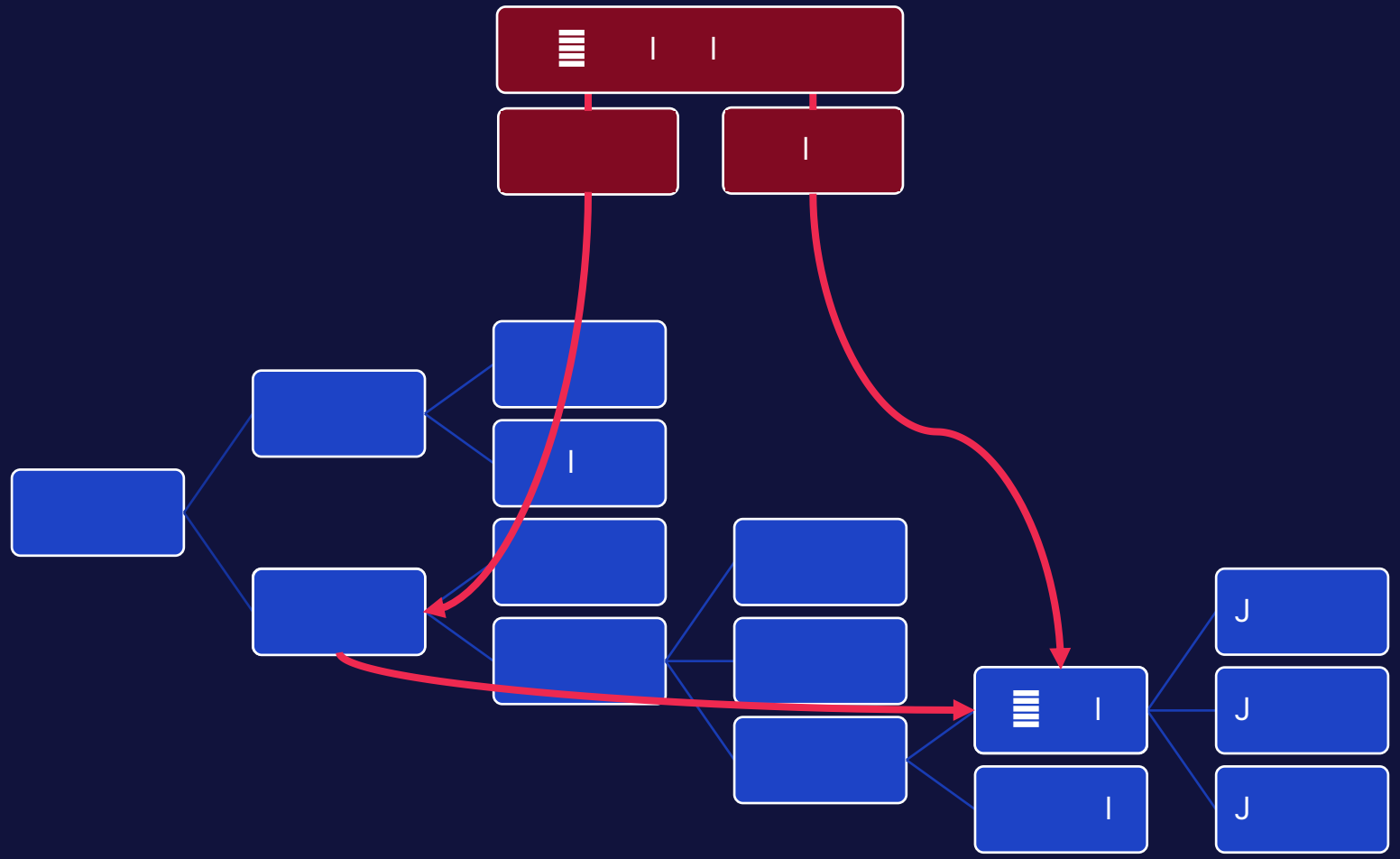
- | | | | |
  |
- |

bouvet

Anatomy of Context



Anatomy of Context



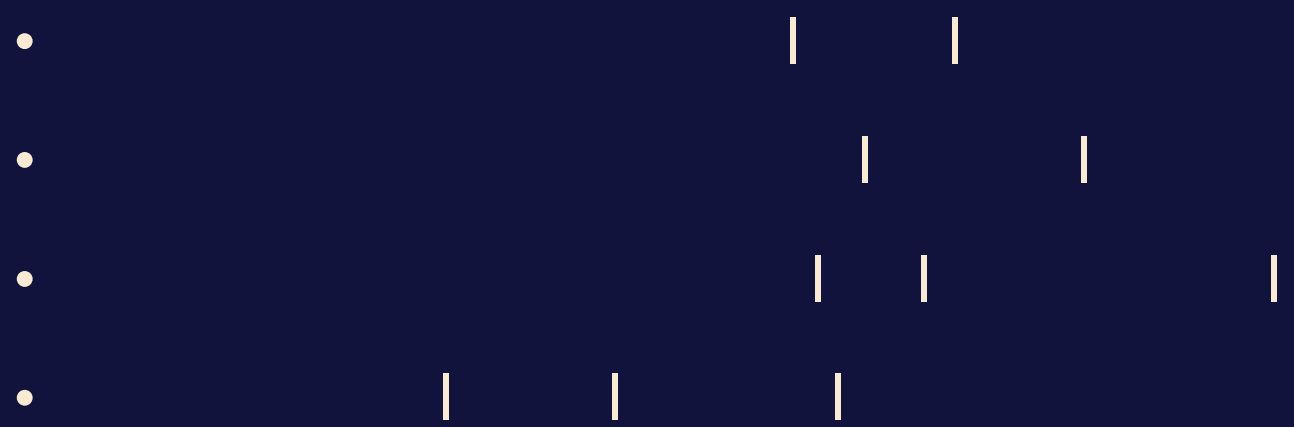
bouvet

UsersContext

-
-

bouvet

<> DisableFieldsContext



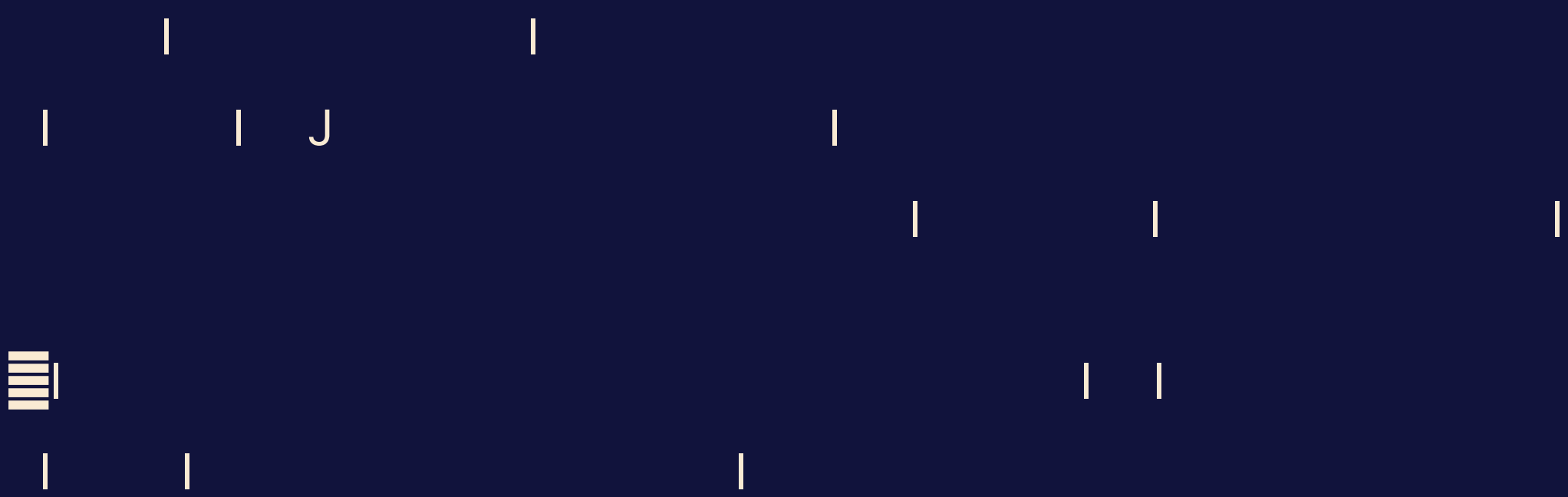
bouvet

Communication with a server

- I I I
- I I
- J I

bouvet

ServerSideUserTable



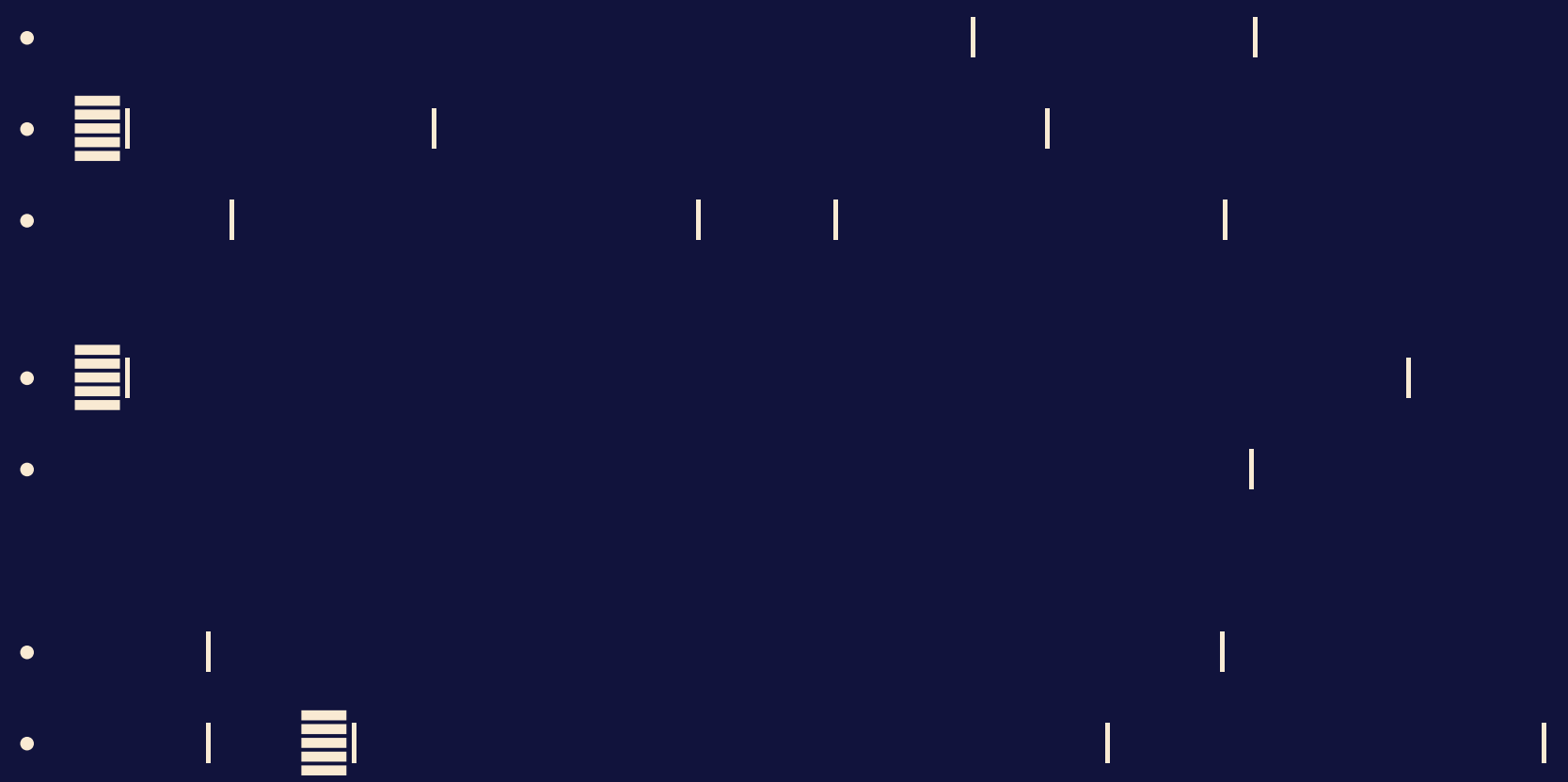
bouvet

Generating clients

-
-
-

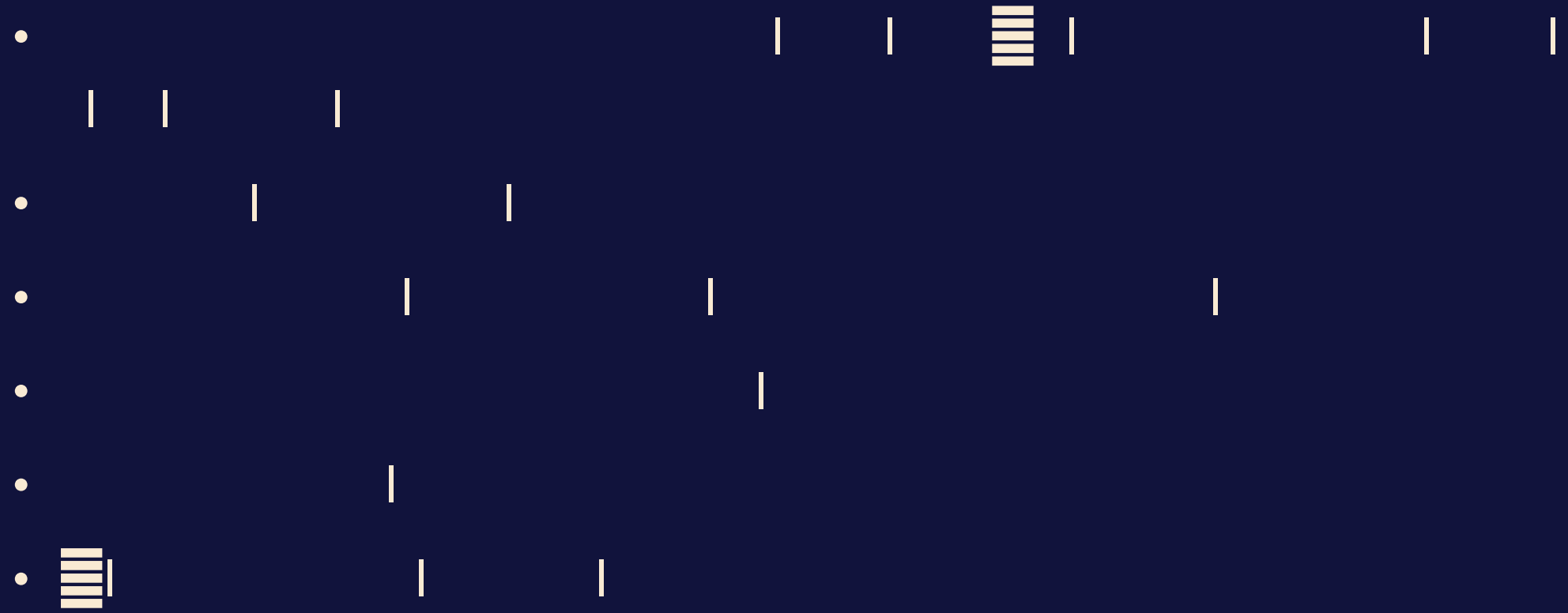
bouvet

<> GroupsTable



bouvet

<> UserDetails



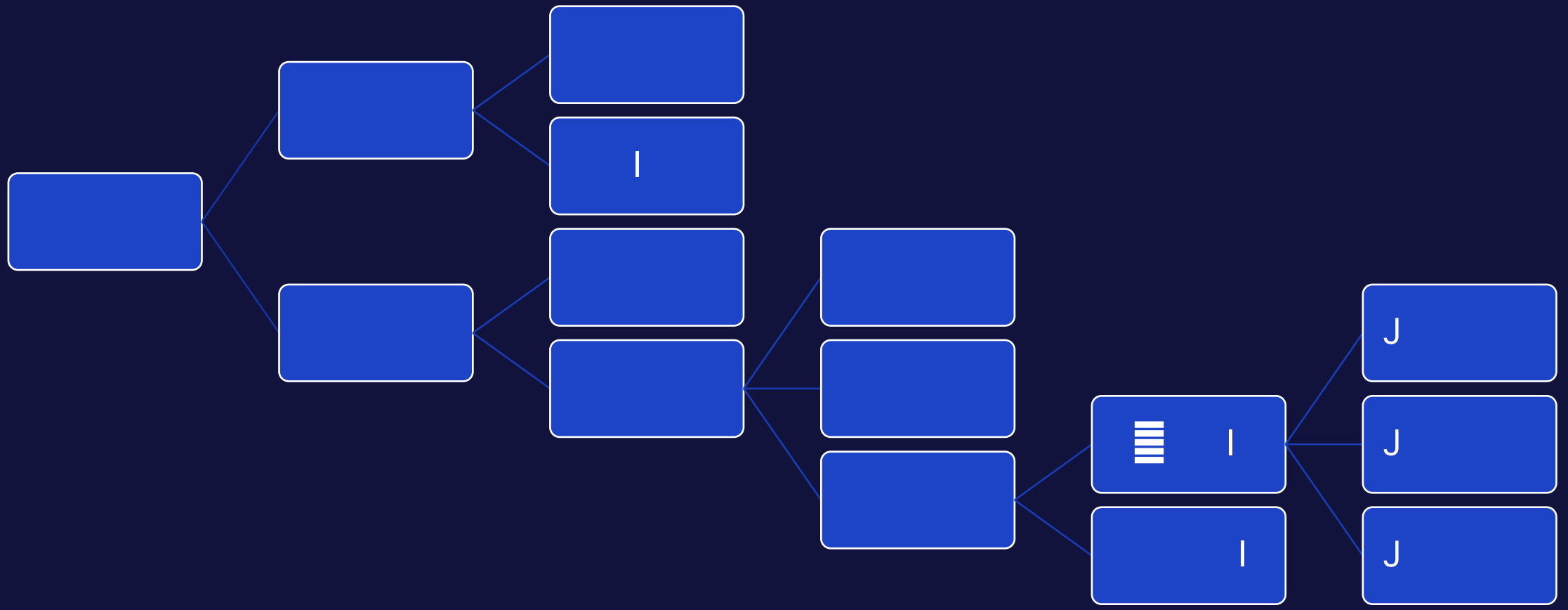
bouvet

Routing

- | | |
- |

bouvet

Anatomy of routing

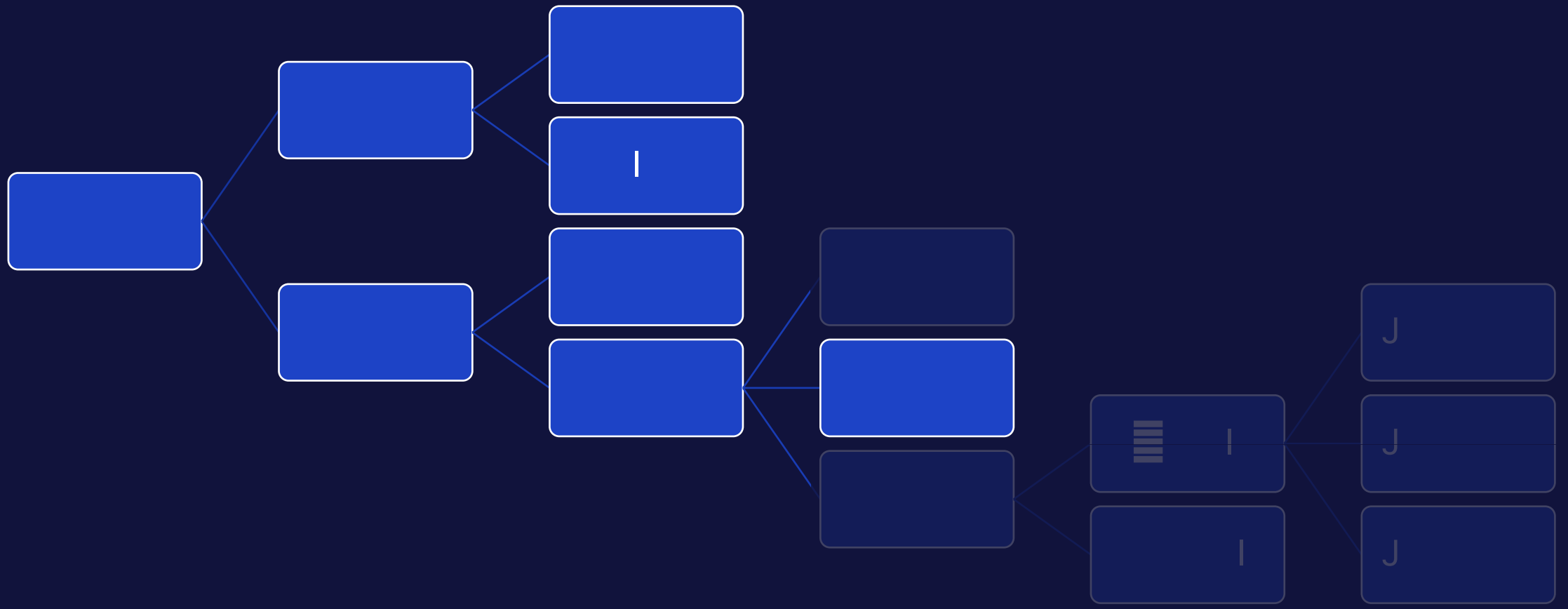


bouvet

Anatomy of

Anatomy of routing

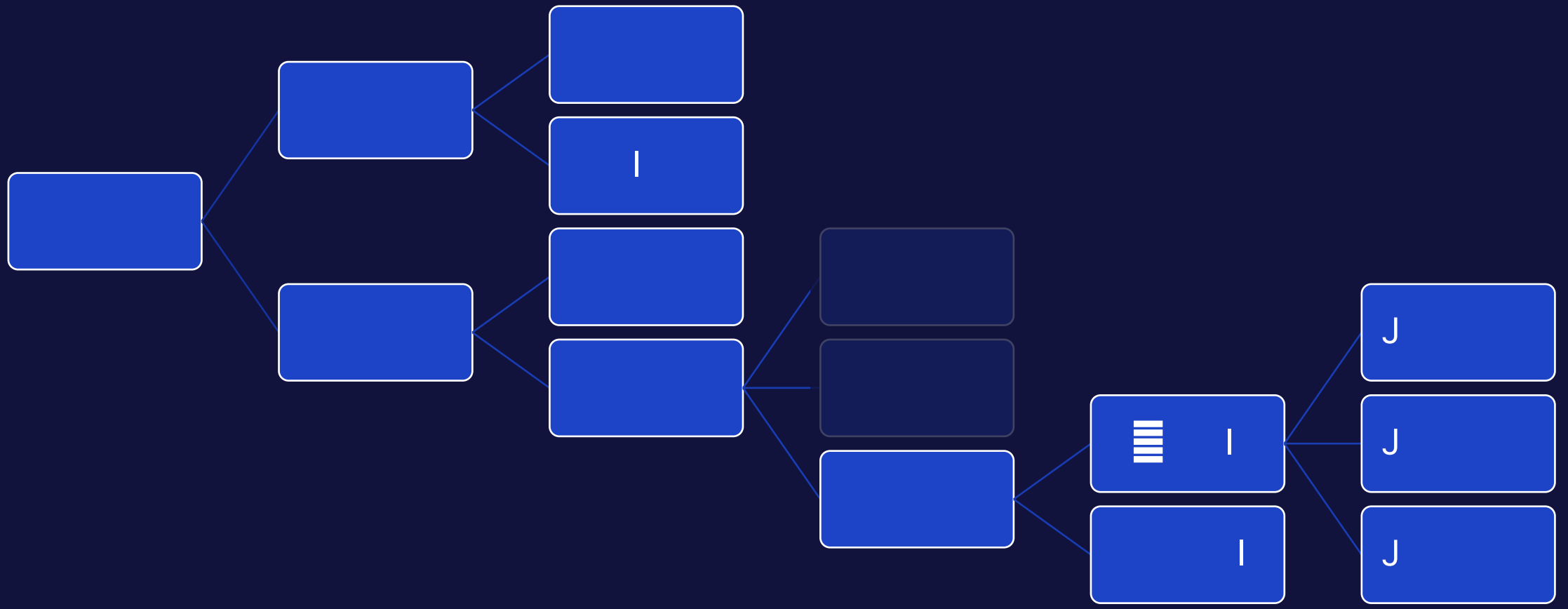
/login



bouvet

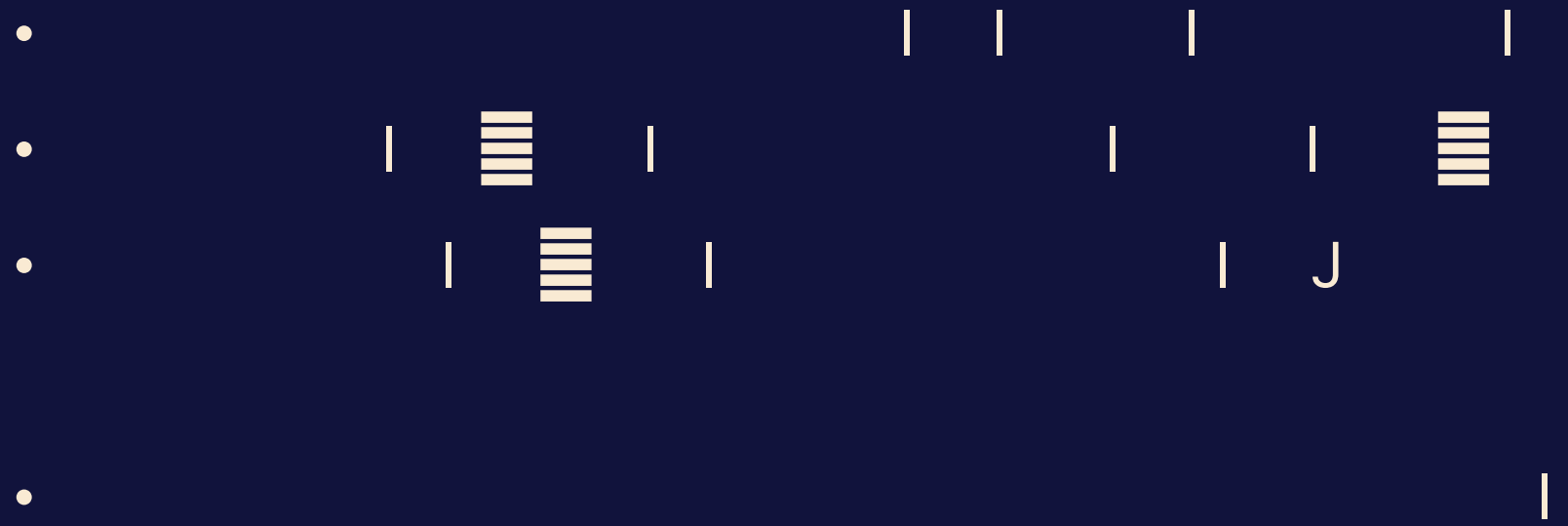
Anatomy of routing

/profile



bouvet

HomePage and UserDetailsPage



bouvet

GroupDetails and Navigation

bouvet

MainLayout

bouvet

<> NoNavigationLayout

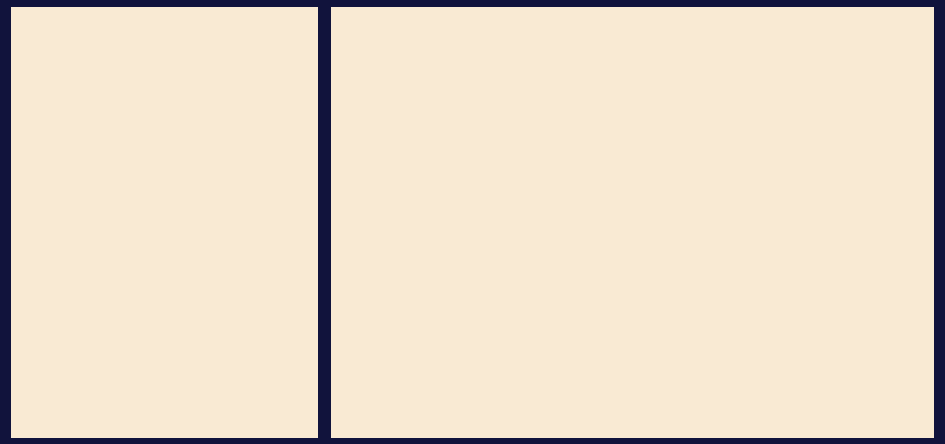
bouvet

Component composition

- | | | | | | |

bouvet

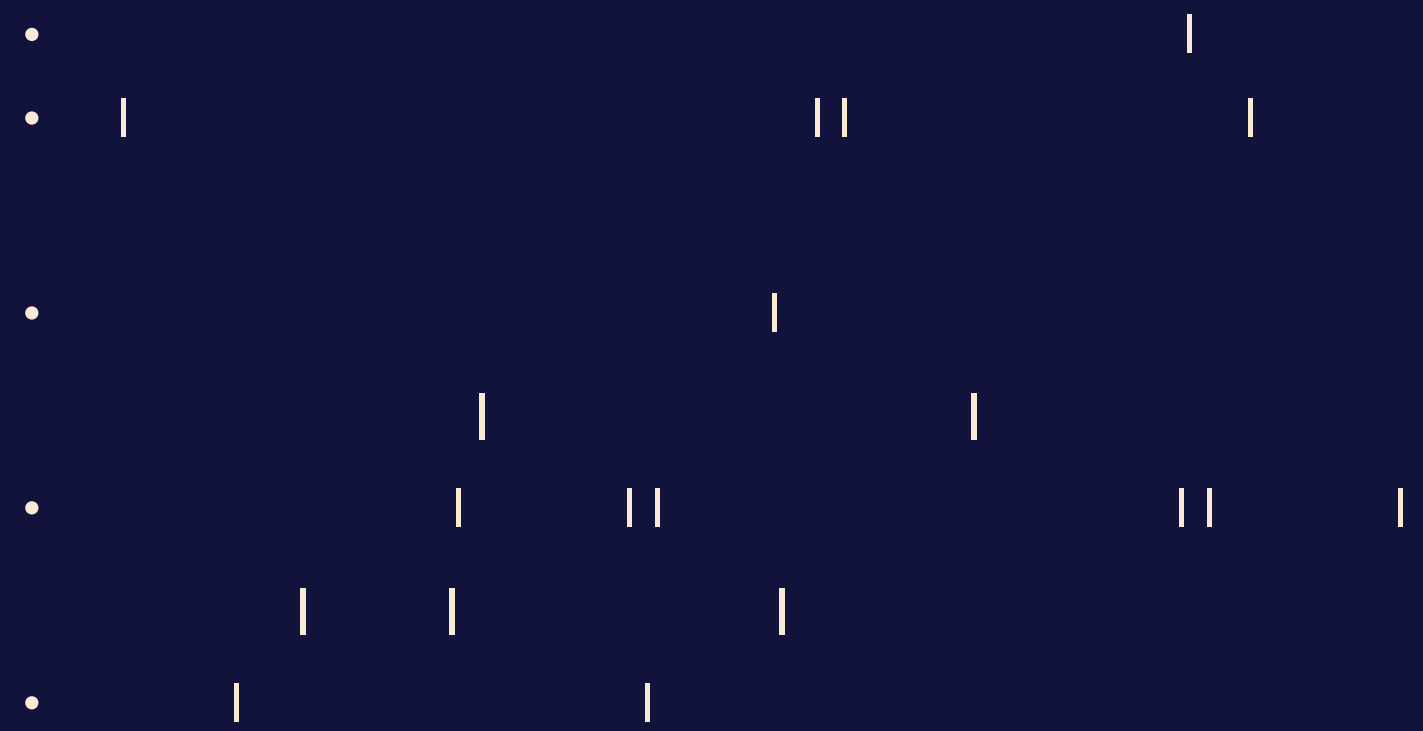
MainSidebar



bouvet

Optimizations

- **Avoid unecessary re-renders**



bouvet

QA

-
- J I

bouvet