
Phero

Hi

Jasper Haggenburg

@Jpunt / @jpunt@mastodon.social

Building stuff since the 90's

Co-founder of **Press Play**

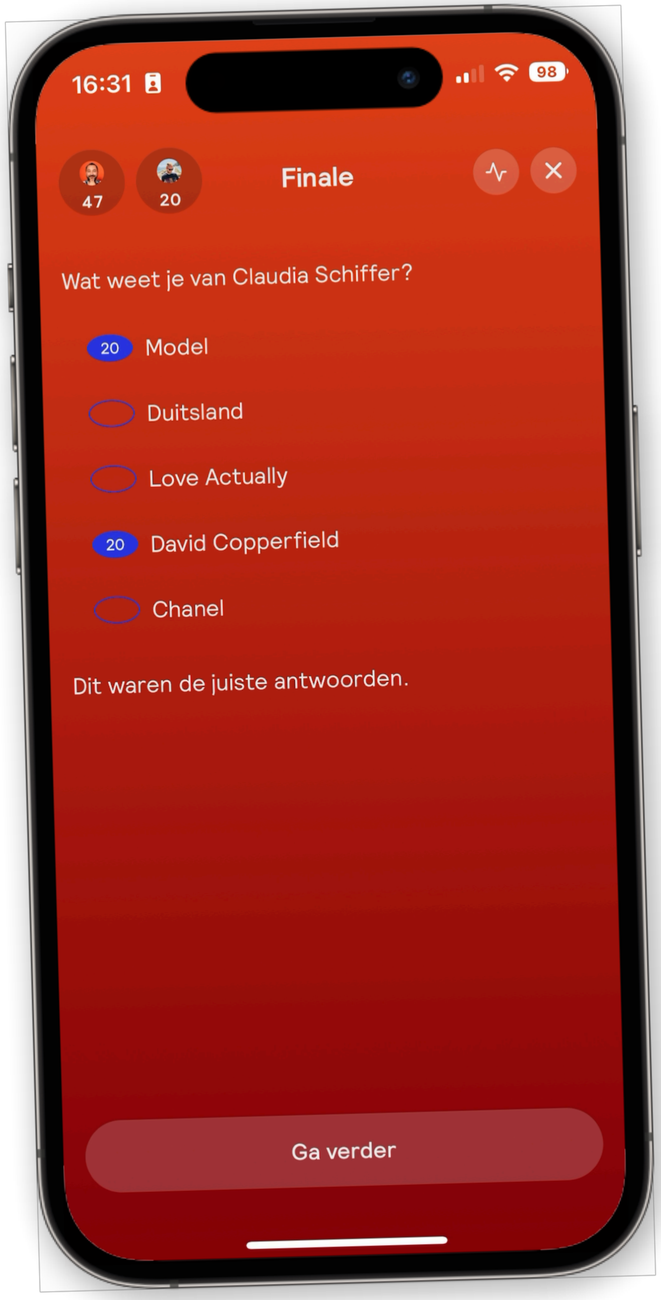
Co-creator of **Phero**

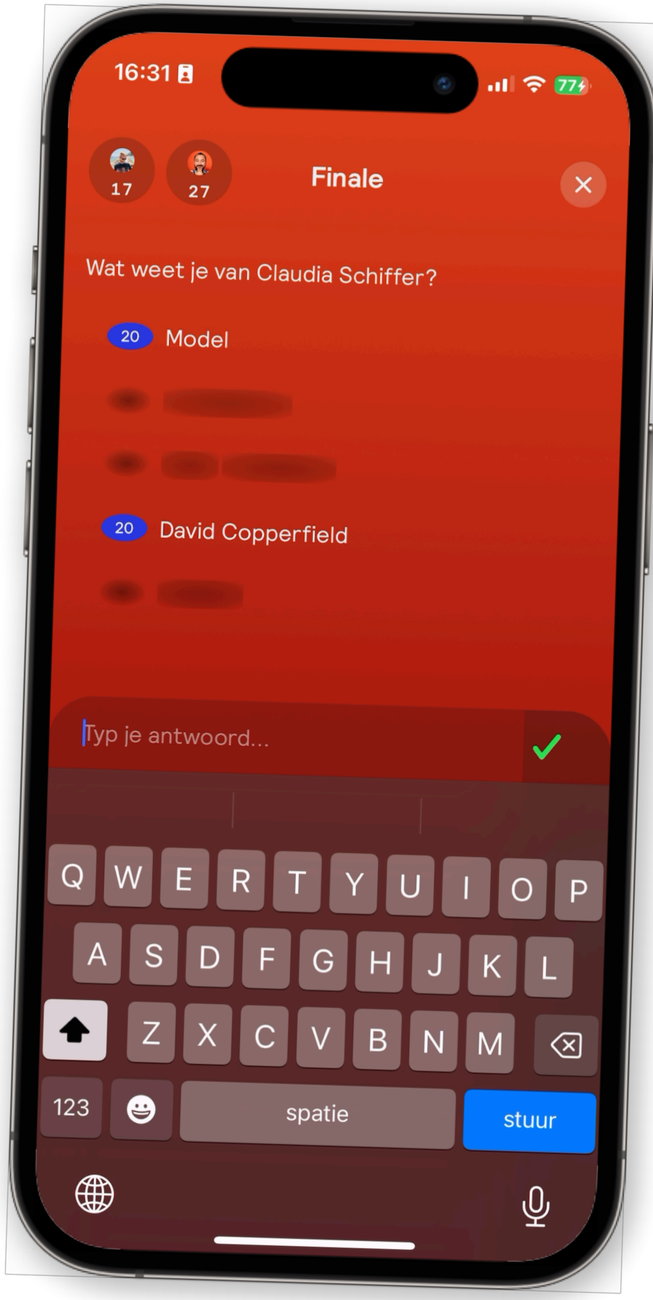

16:55
HI JASPER
THE WAY OF WATER CONNECTS ALL THINGS
TOP 20
SHOW ALL
AVATAR
WAKANDA FOREVER
EXPLORE LOCATIONS
SHOW ALL
MY LOCATION
2 PEOPLE
4 PEOPLE
7 PEOPLE
Home
Anything
Anywhere
Last min
More

team
Press Play

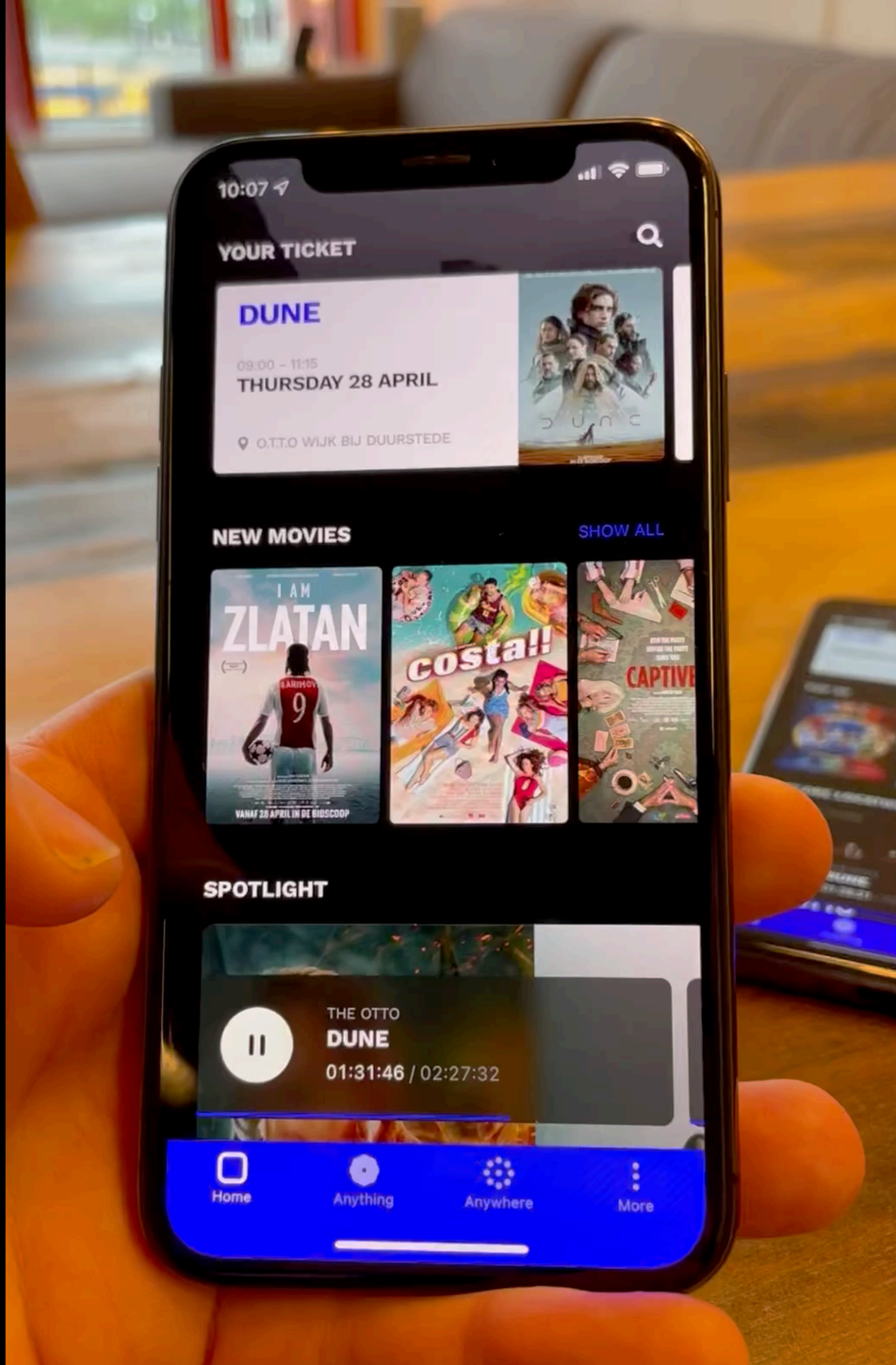
?

you?


16:31
98
47
20
Finale
Wat weet je van Claudia Schiffer?
20 Model
Duitsland
Love Actually
20 David Copperfield
Chanel
Dit waren de juiste antwoorden.
Ga verder


16:31
77
17
27
Finale
Wat weet je van Claudia Schiffer?
20 Model
20 David Copperfield
Typ je antwoord...
spatie
stuur


IMAGINE
PLAY
WATCH
ANYTHING
ANYWHERE
ANYTIME
SCAN ME AND
DOWNLOAD THE APP
App Store


10:07
YOUR TICKET
DUNE
09:00 – 11:15
THURSDAY 28 APRIL
O.T.T.O WIJK BIJ DUURSTEDE
NEW MOVIES
SHOW ALL
I AM
ZLATAN
VANAF 28 APRIL IN DE BIOSCOOP
costa!!
CAPTIVE
SPOTLIGHT
THE OTTO
DUNE
01:31:46 / 02:27:32
Home
Anything
Anywhere
More


Safari File Edit View History Bookmarks Develop Window Help
Mobile App (3:47) 100% 16:22
develop--nvoo-production.netlify.app
Over ons
Programma
Kennisbank
Community
Nederlandse Vereniging voor Overgewicht en Obesitas
Wij zijn een officiële patiëntenvereniging voor iedereen met overgewicht en obesitas in Nederland. Ons doel is om patiënten samen te brengen, de belangen te behartigen, informatie te voorzien en als allerbelangrijkste om ervaringen en problemen veilig te delen!
Onze leden

```
/doc/:id
/search?q={q}
/login
```

```
/doc/:id
/search?q={q}
/login
```

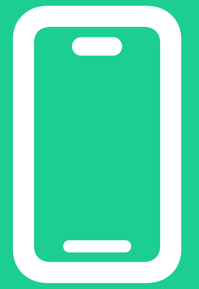
```
interface Doc {
  id: string
  author: User
  content: Component[]
}
```

```
/doc/:id
/search?q={q}
/login
```

```
interface Doc {
  id: string
  author: User
  content: Component[]
}

interface User {
  id: string
  name: string
  email?: string
}

interface TextComponent {
  type: 'text',
  content: string
}

interface ImageComponent {
  type: 'image',
  src: string
}

type Component =
  | TextComponent
  | ImageComponent
```

```
/doc/:id

interface Doc {
  id: string
  author: User
  content: Component[]
}
```

```
/doc/:id

interface Doc {
  id: string
  author: User
  content: Component[]
}
```



```
/doc/:id

interface Doc {
  id: string
  author: User
  content: Component[]
}
```

v1.0.1

v1.0.2

v1.0.3

```
/doc/:id

interface Doc {
  id: string
  author: User
  content: Component[]
}
```

v1.0.1

v1.0.2

v1.0.3

```
/doc/:id

interface Doc {
  id: string
  author: User
  content: Component[]
}
```

v1.0.1

v1.0.2

v1.0.3

```
/doc/:id

interface Doc {
  id: string
  author: User
  content: Component[]
  title: string
}
```

v1.0.1

v1.0.2

v1.0.3

```
/doc/:id

interface Doc {
  id: string
  author?: User
  content: Component[]
  title: string
}
```

v1.0.1

v1.0.2

v1.0.3

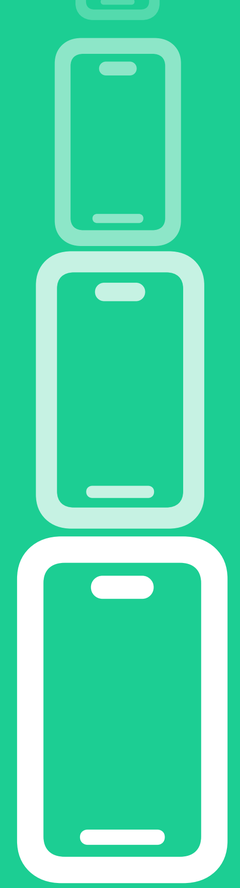
```
/doc/:id

interface Doc {
  id: string
  author?: User
  content: Component[]
  title: string
}
```

v1.0.1

v1.0.2

v1.0.3

```
/doc/:id

interface Doc {
  id: string
  author?: User
  content: Component[]
  title: string
}
```



v1.0.2

v1.0.3

v1.0.4

```
/doc/:id

interface Doc {
  id: string
  author?: User
  content: Component[]
  title: string
}
```

v1.0.1

v1.0.2

v1.0.3

v1.0.4

v1.0.3

```
/doc/:id

interface Doc {
  id: string
  author?: User
  content: Component[]
  title: string
}

interface User {
  id: string
  name: string
  email?: string
}
```

v1.0.3

```
/doc/:id

interface Doc {
  id: string
  author?: User
  content: Component[]
  title: string
}

interface User {
  id: string
  name: string
  email?: string
}
```

```
const response = await fetch(`/docs/${id}`)
const json = await response.json() as Doc
```

```
/doc/:id

interface Doc {
  id: string
  author?: User
  content: Component[]
  title: string
}

interface User {
  id: string
  name: string
  email?: string
}
```

v1.0.3

```
const response = await fetch(`/docs/${id}`)
const json = await response.json() as Doc

// ...

function App() { /* ... */ }
function Navigation() { /* ... */ }
function SettingScreen() { /* ... */ }
function Doc() {
  return (
    <View>
    </View>
  )
}
```

```
/doc/:id

interface Doc {
  id: string
  author?: User
  content: Component[]
  title: string
}

interface User {
  id: string
  name: string
  email?: string
}
```

v1.0.3

```
const response = await fetch(`/docs/${id}`)
const json = await response.json() as Doc

// ...

function App() { /* ... */ }
function Navigation() { /* ... */ }
function SettingScreen() { /* ... */ }
function Doc() {
  return (
    <View>
      <Text>Hey there, {doc.author.name}!</Text>
    </View>
  )
}
```

v1.0.3

```
/doc/:id

interface Doc {
  id: string
  author?: User
  content: Component[]
  title: string
}

interface User {
  id: string
  name: string
  email?: string
}
```

```
const response = await fetch(`/docs/${id}`)
const json = await response.json() as Doc

// ...

function App() { /* ... */ }
function Navigation() { /* ... */ }
function SettingScreen() { /* ... */ }
function Doc() {
  return (
    <View>
      <Text>Hey there, {doc.author.name}!</Text>
    </View>
  )
}
```

```
/doc/:id

interface Doc {
  id: string
  author?: User
  content: Component[]
  title: string
}

interface User {
  id: string
  name: string
  email?: string
}
```

v1.0.3

```
const response = await fetch(`/docs/${id}`)
const json = await response.json() as Doc

// ...

function App() { /* ... */ }
function Navigation() { /* ... */ }
function SettingScreen() { /* ... */ }
function Doc() {
  return (
    <View>
      <Text>Hey there, {doc.author.name}!</Text>
    </View>
  )
}
```

```
/doc/:id

interface Doc {
  id: string
  author?: User
  content: Component[]
  title: string
}

interface User {
  id: string
  name: string
  email?: string
}
```

v1.0.3

```
const response = await fetch(`/doc/${id}`)
const json = await response.json() as Doc

// ...

function App() { /* ... */ }
function Navigation() { /* ... */ }
function SettingScreen() { /* ... */ }
function Doc() {
  return (
    <View>
      <Text>Hey there, {doc.author.name}!</Text>
    </View>
  )
}
```

```
/doc/:id

interface Doc {
  id: string
  author?: User
  content: Component[]
  title: string
}
```

How can we all get along?*

** frontend & backend in this case*

How can we all get along?*

** frontend & backend in this case*

- **F*ck it / DIY**

How can we all get along?*

** frontend & backend in this case*

- F*ck it / DIY

- **GraphQL**

How can we all get along?*

** frontend & backend in this case*

- F*ck it / DIY

- **GraphQL**

```
type Doc {
  id: ID!
  author: User
  content: Component[]
  title: String!
}

type User {
  id: ID!
  name: String!
  email: String
}
```

How can we all get along?*

** frontend & backend in this case*

- F*ck it / DIY

- **GraphQL + Codegen**

```
type Doc {
  id: ID!
  author: User
  content: Component[]
  title: String!
}

type User {
  id: ID!
  name: String!
  email: String
}
```

How can we all get along?*

** frontend & backend in this case*

- F*ck it / DIY

- GraphQL + Codegen

- **tRPC + ZOD**

How can we all get along?*

** frontend & backend in this case*

- F*ck it / DIY

- GraphQL + Codegen

- **tRPC + ZOD**

```
const Doc = z.object({
  id: z.string(),
  author: z.instanceof(User).optional(),
  content: z.array(Component),
  title: z.string()
})
type Doc = z.infer<typeof Doc>

const User = z.object({
  id: z.string(),
  name: z.string(),
  email: z.string().optional()
})
type User = z.infer<typeof User>
```

How can we all get along?*

** frontend & backend in this case*

- F*ck it / DIY

- GraphQL + Codegen

- tRPC + ZOD

- ***Phero***

How can we all get along?*

** frontend & backend in this case*

- F*ck it / DIY
- GraphQL + Codegen
- tRPC + ZOD
- ***Phero***

```
interface Doc {
  id: string
  author?: User
  content: Component[]
  title: string
}

interface User {
  id: string
  name: string
  email?: string
}
```


Demo

→ https://phero.dev

→ https://pressplay.dev

→ @PheroHQ

→ @phero_dev@mastadon.social