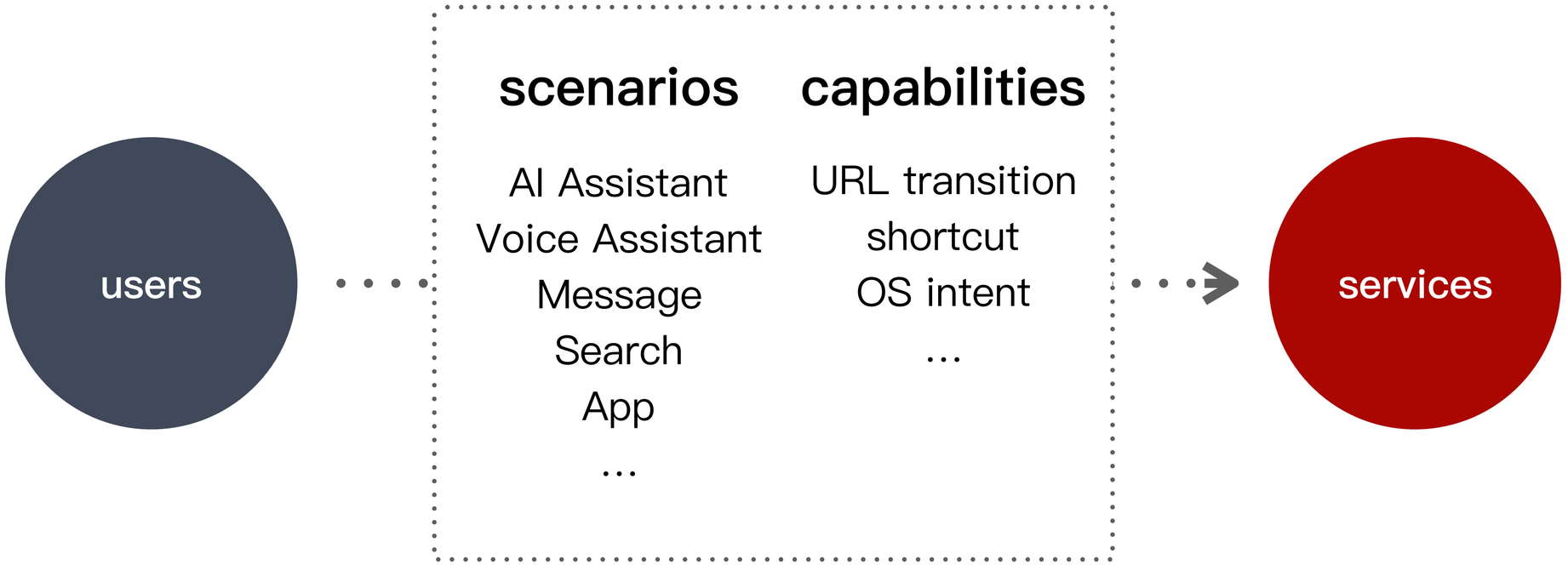
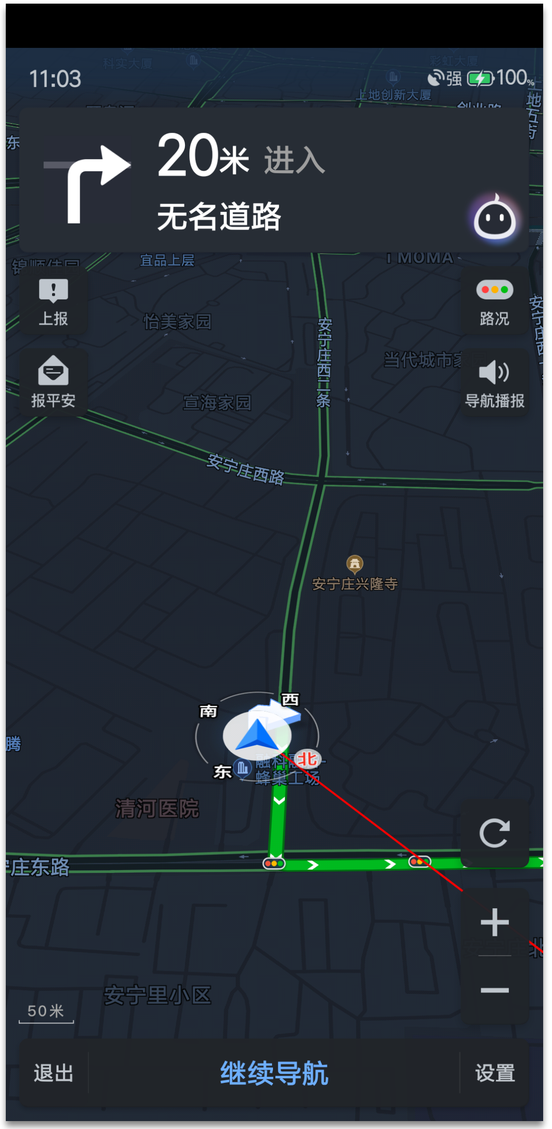
MiniApp Widgets

the OS display and connect many services



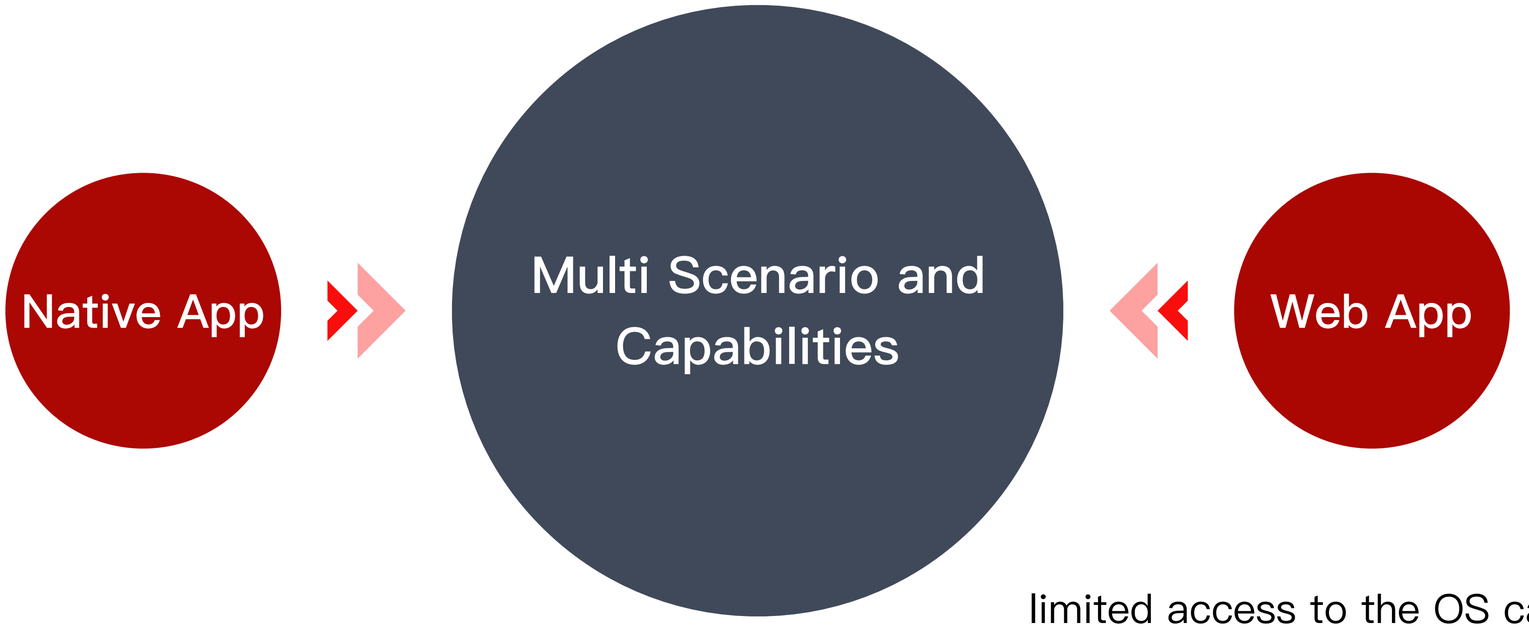
AI Assistant

navigation service



scenario + capability = service
–> an organic OS

1. Use Widget to display content of the service for each scenario
2. navigate to the webpage/app page for user interaction
3. share data with the Widget

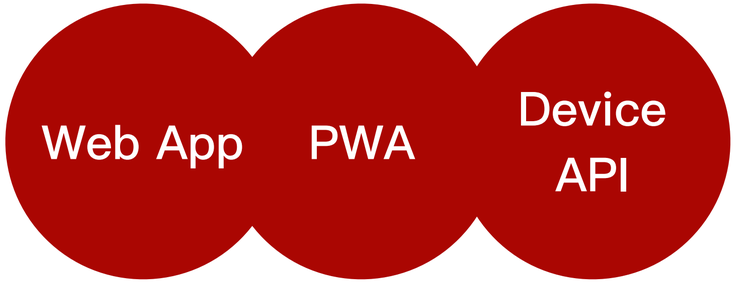


wnload+install before usage, the user intent gets disrupted

Native Widget works on desktop, with limited functions

limited access to the OS capabilities

not enough components or rich APIs



design for browsers

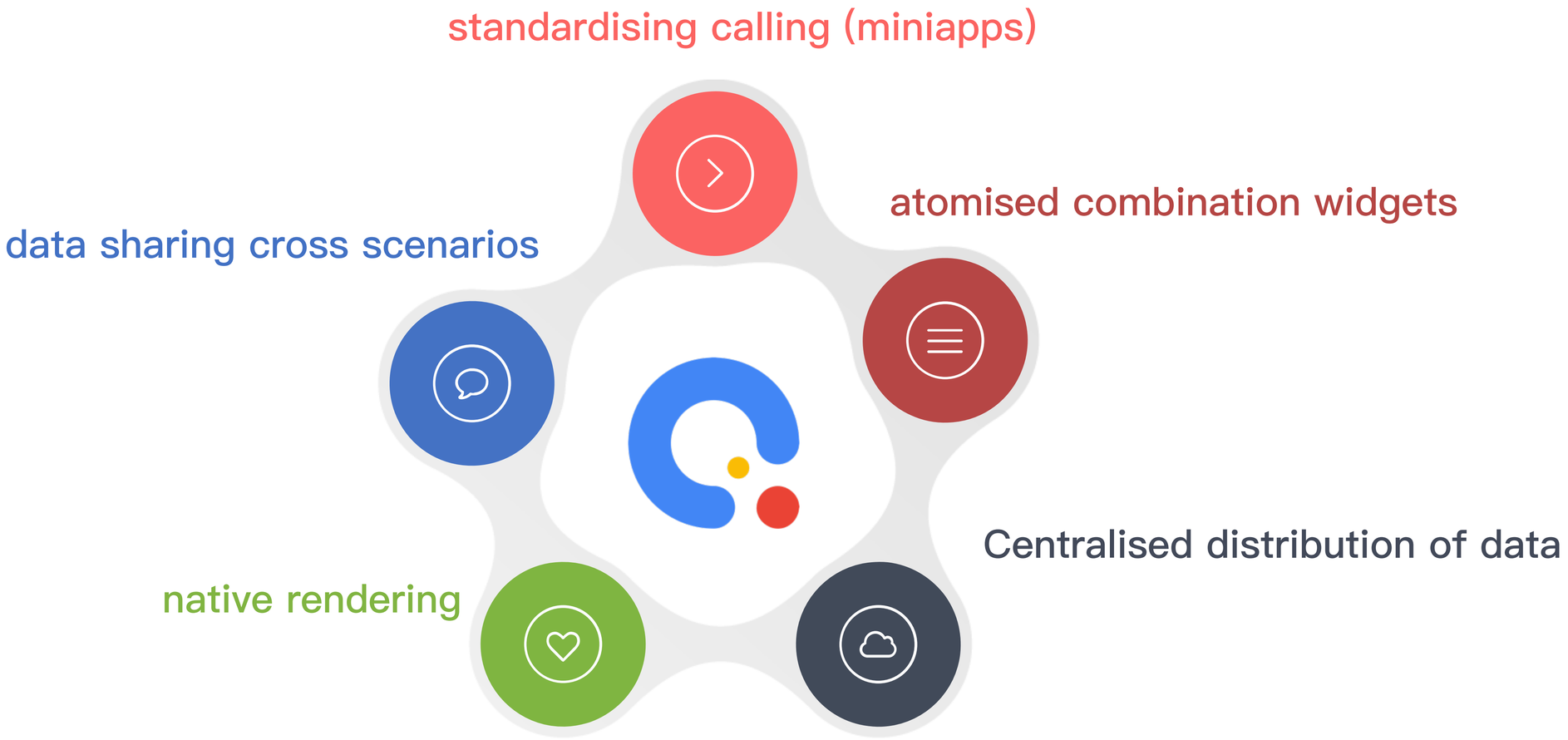
Problems:

1. security issues for data distribution
2. difficult to share data when a service is cross scenarios
3. no standardised way to share data
4. no standardised way to call a webpage/apppage of a micro service

data sharing
cross scenario

calling a webpage
for a micro service

design for multi scenarios



Demo of QuickApp

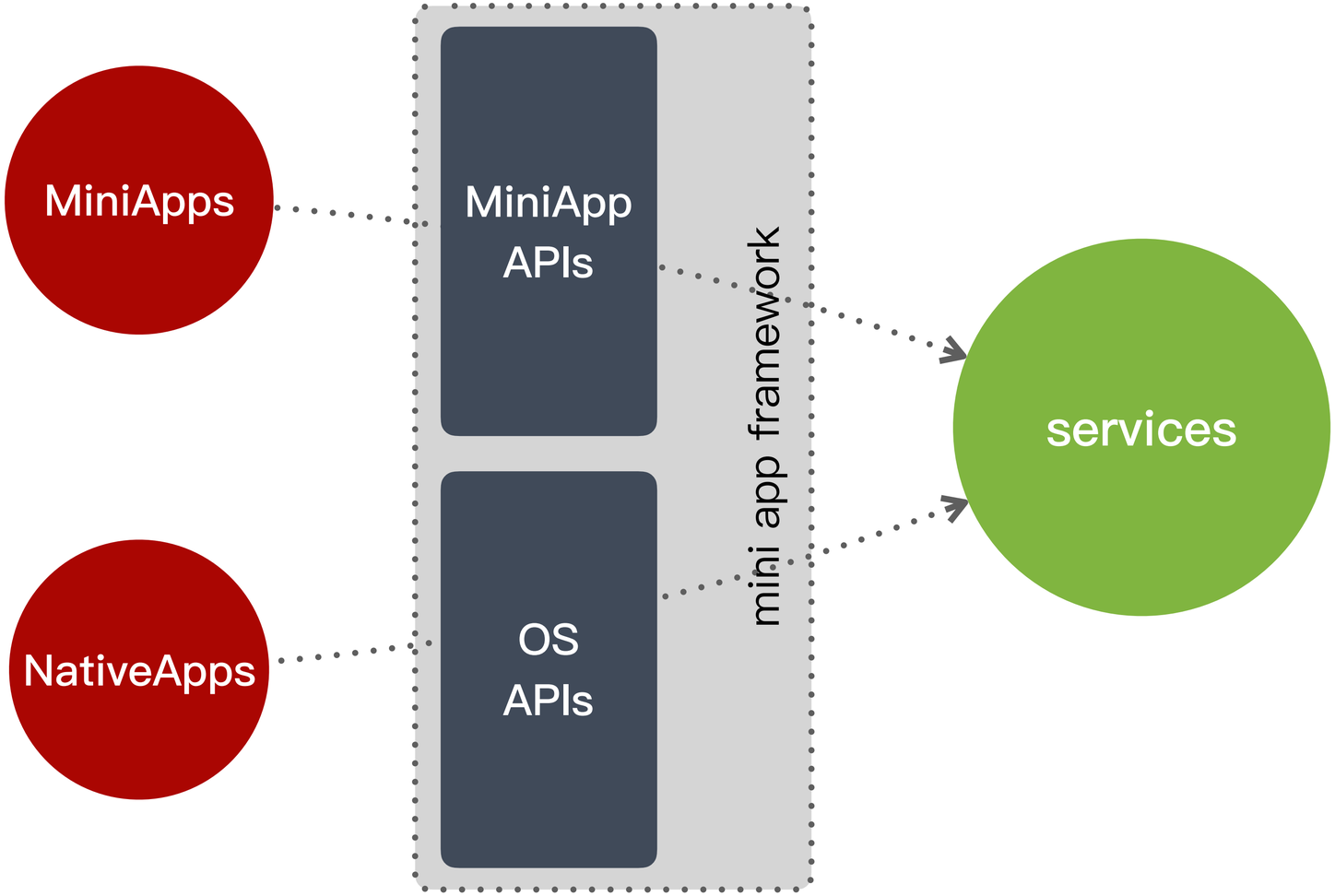
1. standardising calling: URL scheme

```
http(s)://hapjs.org/app/<package>/[path][?key=value]
hap://app/<package>/[path][?key=value]
```

2. data processing in the detail page

calling according to the hidden intent

some miniapps provide a centralised platform to analysis the APIs and OS, and process the hidden the intent of navigation, filter, or sharing...



MiniApp Widget atomise the centent, and combine them on detail pages

programming for a widget

1. each widget is describe by URL
2. rich APIs and Capabilities
3. multiple ways to authorise

combination according to the scenarios

1. Centralised distribution and update
2. rendering in the webview/native view
3. Data Isolation among widgets

Data Sharing cross Scenarios

1. only one instance of a MiniApp inside the whole OS
2. data sharing between widgets and miniapps
3. scenarios and miniapps can share data under certain control

“login once, works everywhere”

design for scenarios

1. standardising calling (miniapps)
2. atomised combination widgets
3. data sharing cross scenarios