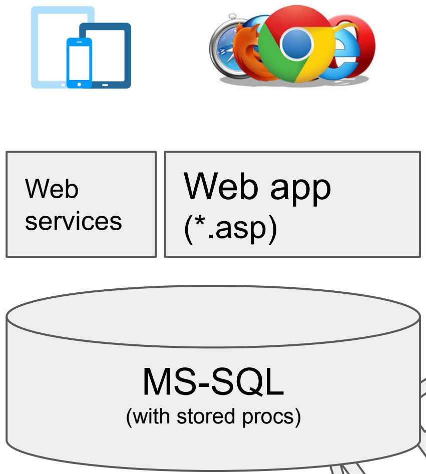
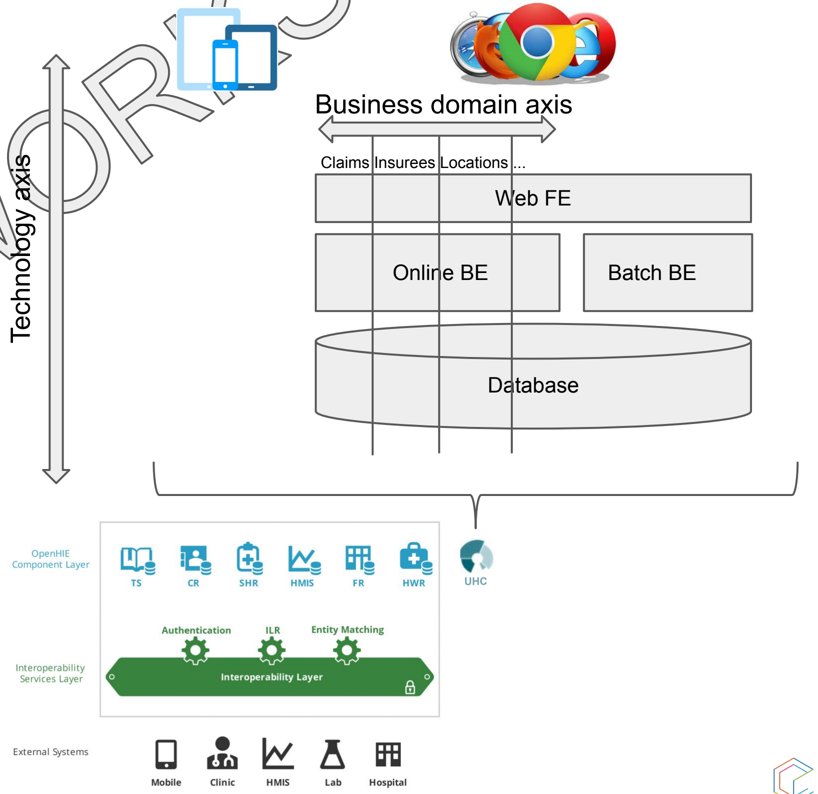
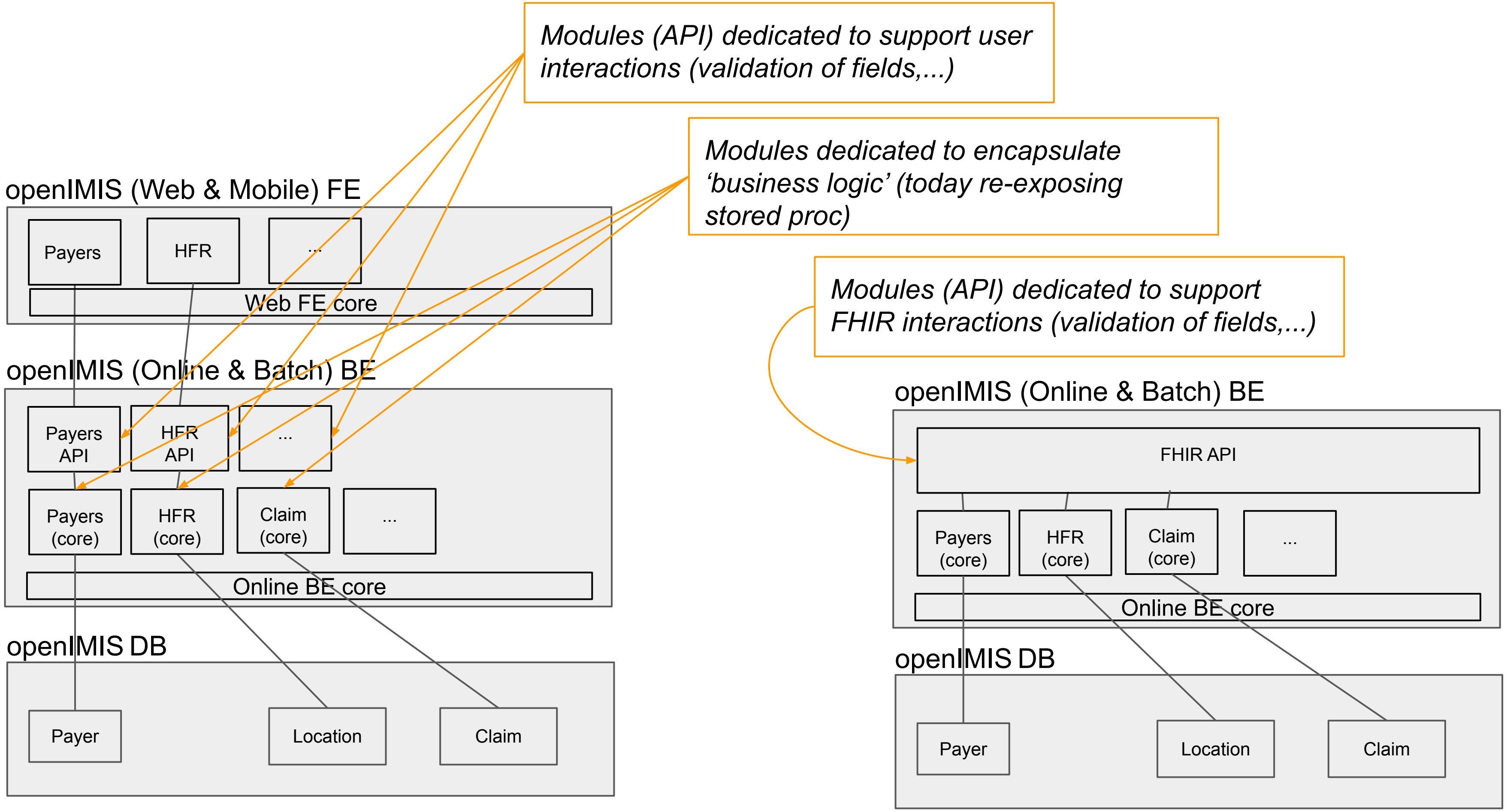
Modularity and openIMIS

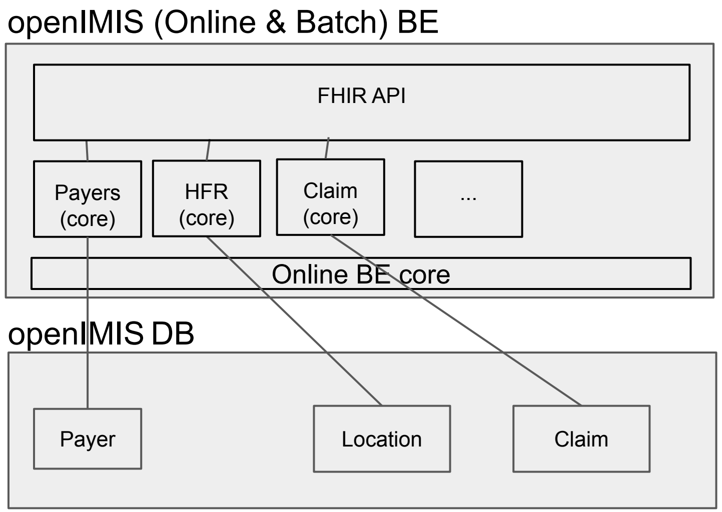
Today openIMIS is “standalone” and “monolithic”

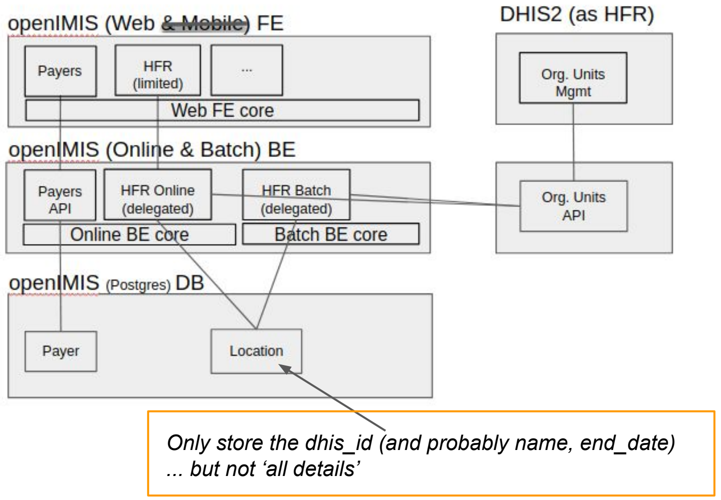


Tomorrow openIMISs are **platforms** integrated into large landscape and assembled from “modules”




Modules (API) dedicated to support user interactions (validation of fields,...)
Modules dedicated to encapsulate 'business logic' (today re-exposing stored proc)
Modules (API) dedicated to support FHIR interactions (validation of fields,...)
openIMIS (Web & Mobile) FE
Payers
HFR
...
Web FE core
openIMIS (Online & Batch) BE
Payers API
HFR API
...
Payers (core)
HFR (core)
Claim (core)
...
Online BE core
openIMIS DB
Payer
Location
Claim
openIMIS (Online & Batch) BE
FHIR API
Payers (core)
HFR (core)
Claim (core)
...
Online BE core
openIMIS DB
Payer
Location
Claim





Only store the dhis_id (and probably name, end_date) ... but not 'all details'

Within Claim (FHIR) resource, when we do

$$\underbrace{\texttt{claim.facility}}_{\text{fhir model}} = \underbrace{\texttt{claim.location.id}}_{\text{openIMIS model}}$$

we 'materialize' a dependency between claim and location
… which is 'normal'.

If, tomorrow, we replace openIMIS location module by an intergration with a HFR management system (where you would be organizing, merging,... HFs) we will:

- simplify UI (only search capabilities)
- not implement some "business logic" (like hierarchical organisation of HFs,...)

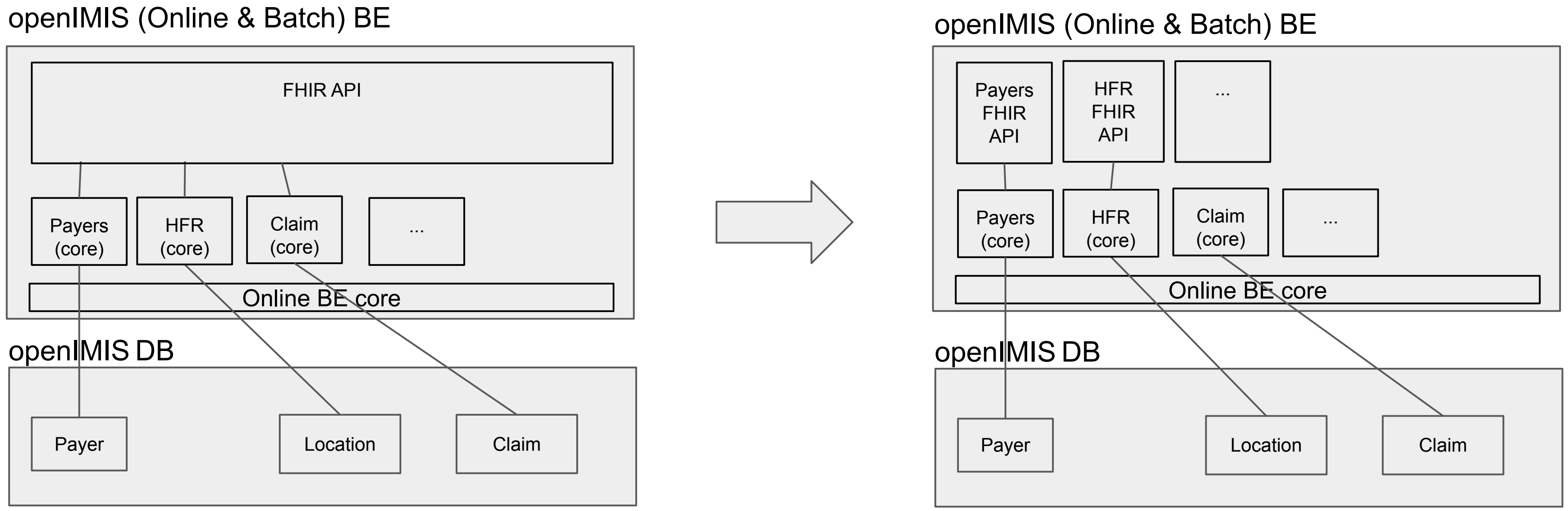
… but we will still need some sort of 'reference' (id) for locations in openIMIS database!

But if we do:

```
claim.facility.end_date = claim.location.expiry
claim.facility.gps = claim.location.gps
```

we 'reinforce' (extend) the dependency (FHIR api for claim will only work if location provides gps coordinates

Breaking FHIR API into (sub)Modules



… gain is to be able to deploy FHIR API per resource...