



Modular Transformation

Bonn workshop: 26-28 / 02



Agenda

- **Modularity**
 - Modularity & openIMIS
 - Challenges of modularity
- **Transition strategy**
- **Target technology stack**
- **Proposed transition phases**



Modularity levels

Solution, Software Component and Entity levels:

<https://openimis.atlassian.net/wiki/spaces/OP/pages/586383361/Target+modular+Architecture>

- Solution level : OpenHIE interop
- Software components level: Plugins, contributions & messaging
- Entity level: entity (and associated screens) customization



Why modularity?

- **Flexibility** (operate in distinct contexts)
- **Focus on** (module encapsulated) **added value** (and delegate to others for what they do best)
- **Ease evolution** (decoupled modules can have distinct lifecycles)
- **Ease “spread” teams collaboration** (ease build of source community)

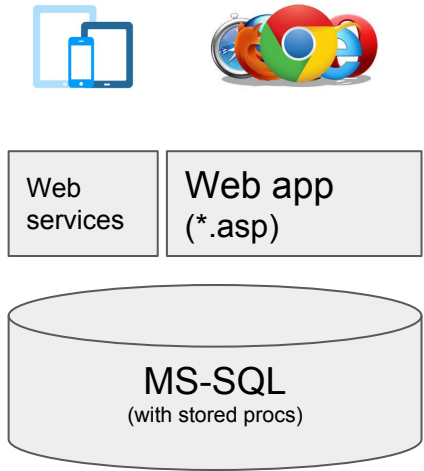
Modularity and “standards” (all specifications, not only talking about FHIR)

- Interoperability: “connectors”, data exchanges
- Encapsulate “experience” of others in the field
- Replaceability (more or less theoretical)
- ... but standards have a cost
 - they constraint the solution landscape (by principle: whatever flexibility/extensibility they claim to support)
 - they *also* address problems you don't have (now)

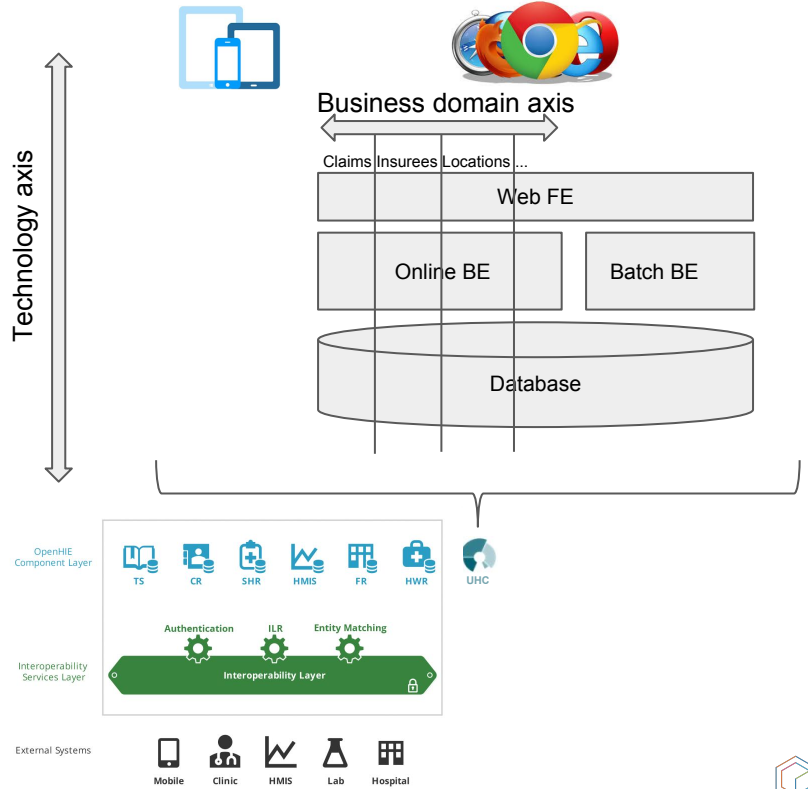


Modularity and openIMIS

Today openIMIS is “standalone” and “monolithic”



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



The cost of modularity

- Modules have to be **loosely coupled**, yet:
 - we have to guarantee **data integrity**
 - we have to guarantee **compatibility**... along modules lifecycles, assembled in various contexts
- Module boundaries is difficult to define (need to anticipate where we will need to “cut”)
... and errors at that step are the most expensive

Coarse grained

- Few dependencies to manage (easy to develop)
- Performances:
 - Optimisation capabilities
 - Less distributed system - friendly
 - Easy to “operate” (install, backup, monitor,...)
- Less flexible / reusable

Fine grained

- Dependencies management (& development constraints)
- Performances:
 - Communications between modules
 - Distributed system - ready
 - Harder to “operate” (install, backup, monitor,...)
- Flexibility (,...)



Modules boundaries “hint” questions

- Is there an existing product that does *‘this’* “for itself” ?
(... and towards which I should be able to build an interface)
- Do I have to (anticipate the need for) provide distinct implementation of *‘this’* in distinct contextes ?
 - ... because user organisation is different
 - ... because intrinsic complexity is different
 - ...
- Are these functionalities enabled/disabled altogether in one context



“Dynamic” (UI/service/...) boundaries

FamilyDAL.vb

```
27 '
28
29 Public Class FamilyDAL
30     Private data As New ExactSQL
31
32     'Corrected
33     Public Sub LoadFamily(ByRef eFamily As IMIS_EN.tblFamilies)
34         Dim data As New ExactSQL
35         Dim sSQL As String = ""
36         sSQL += " SELECT I.InsureeId, I.CHFID, I.LastName, I.OtherNames, I.DOB, I.Phone, I.isOffline InsureeIsOffline, I.Educat
37         sSQL += " F.Poverty, F ConfirmationType, R.RegionId, R.RegionName, D.DistrictName, D.DistrictId, V.VillageId, V.Village
38         sSQL += " I.CurrentAddress, I.CurrentVillage , I.HFID, HF.LocationId FSPDistrictId, HF.HFCareType, F.FamilyType,"
39         sSQL += " F.FamilyAddress, F.Ethnicity,F ConfirmationNo, F.ValidityTo, F.isOffline"
40         sSQL += " from tblFamilies F"
41         sSQL += " INNER JOIN tblVillages V ON V.VillageId = F.LocationId"
42         sSQL += " INNER JOIN tblWards W ON W.WardId = V.WardId"
43         sSQL += " INNER JOIN tblDistricts D ON D.DistrictId = W.DistrictId"
44         sSQL += " INNER JOIN tblRegions R ON R.RegionId = D.RegionId"
45         sSQL += " INNER JOIN tblInsuree i ON f.FamilyID = i.FamilyID"
46         sSQL += " LEFT OUTER JOIN tblHF HF ON HF.HFID = I.HFID"
47         sSQL += " WHERE F.FamilyId = @FamilyId"
48
49         data.setSqlCommand(sSQL, CommandType.Text)
50
51         data.params("@FamilyId", SqlDbType.Int, eFamily.FamilyID)
52         Dim dr As DataRow = data.Filldata()(0)
53         Dim eInsurees As New IMIS_EN.tblInsuree
```



Keeping dependencies under control

- Prevent bi-directional dependencies (... and cyclic dependencies!)
 - Dependency Inversion Principle



- Open/Close Principle
 - Extension Points
 - Composition (over inheritance)
- Loosely coupling
 - Events and pub-sub and/or mapped services (internal bus, ESB,...)

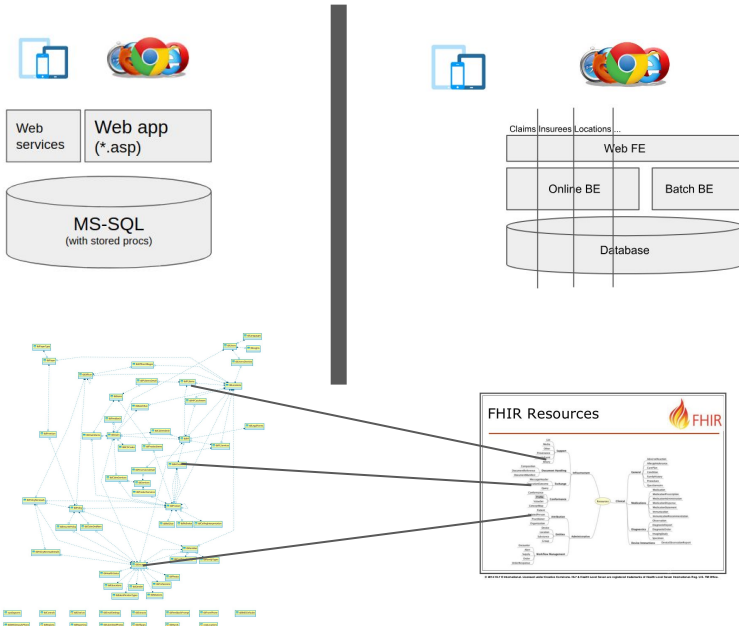
Agenda

- **Modularity**
 - Modularity & openIMIS
 - Challenges of modularity
- **Transition strategy**
- **Target technology stack**
- **Proposed transition phases**



Transition strategies

“One shot” (rebuild + data migration)



“Iterative and incremental”



Agenda

- **Modularity**
 - Modularity & openIMIS
 - Challenges of modularity
- **Transition strategy**
- **Target technology stack**
- **Proposed transition phases**



Technology stack

Choice criteria:

- Open Source based, Open Sourced and Open Source “friendly”
- “Main stream” and “future proof”
- Well documented and easy to access (learn / ...) ⇒ easy to contribute to openIMIS

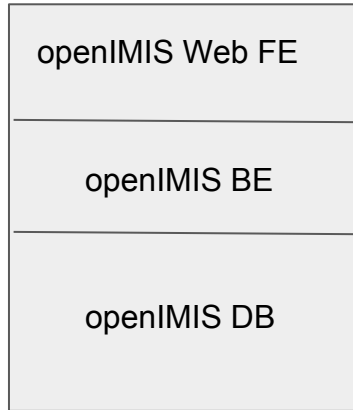
Open Source known difficulties

- No ‘authority’ to guide *our* choices
... and take the responsibility of the coherence of the whole
- More “volatile” (subject to ‘hypes’)
- Less (guided) ‘transition’ between components

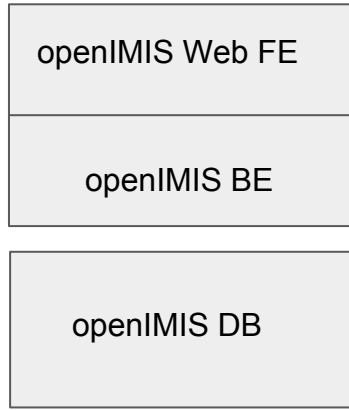


Containerization: deployment “à la carte”

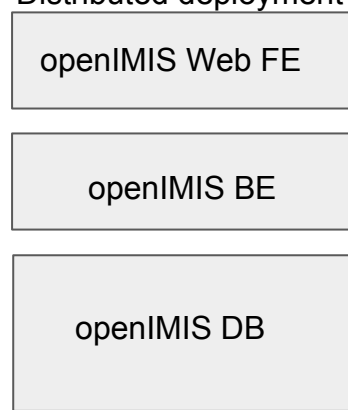
openIMIS -
Monolithic instance



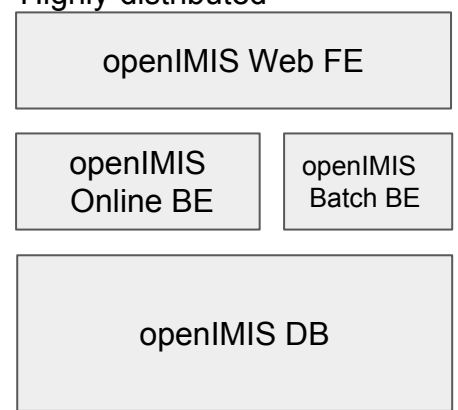
openIMIS -
Dedicated DB instance



openIMIS -
Distributed deployment



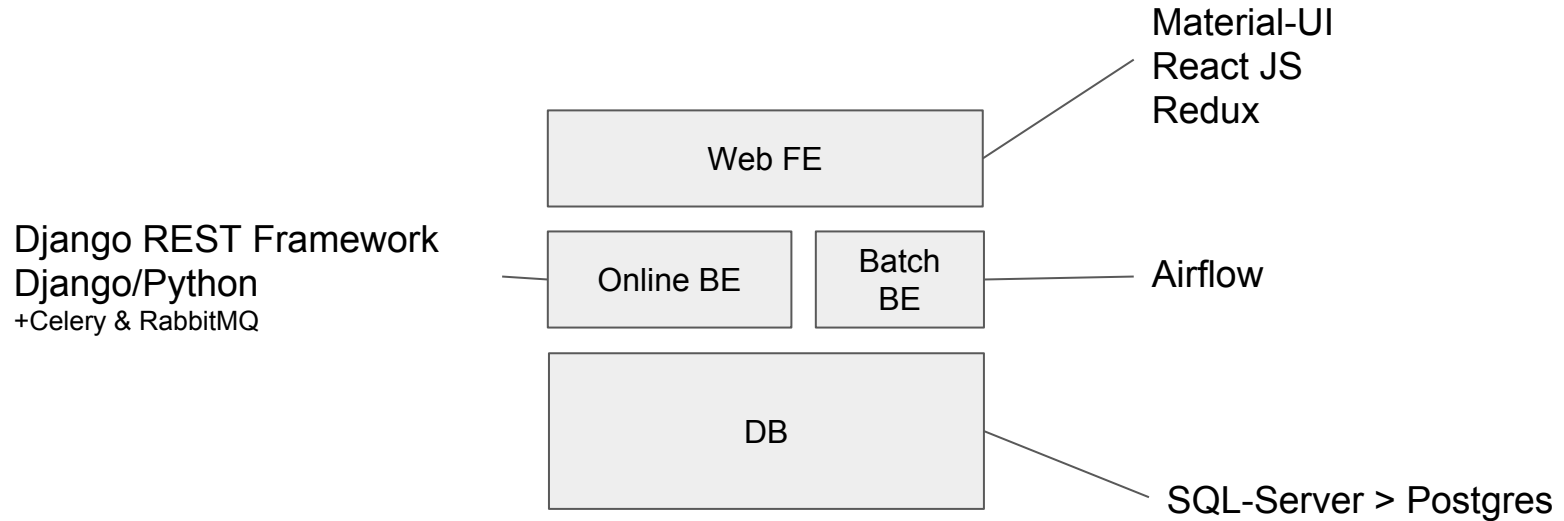
openIMIS -
Highly-distributed



Scale UP



Technology stack



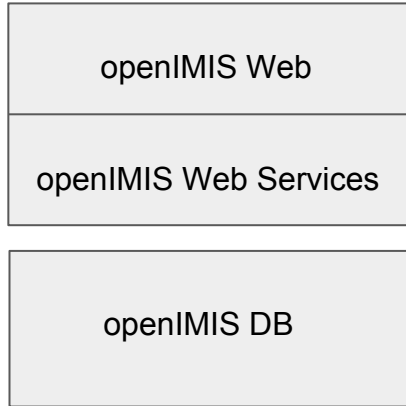
Agenda

- **Modularity**
 - Modularity & openIMIS
 - Challenges of modularity
- **Transition strategy**
- **Target technology stack**
- **Proposed transition phases**

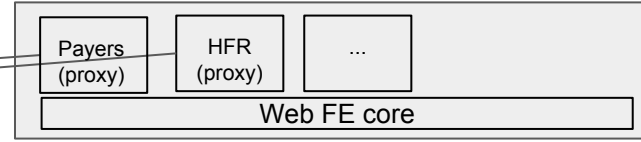


Phase 1 : all proxies

openIMIS (current)



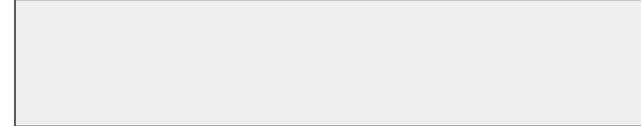
openIMIS (Web & Mobile) FE



openIMIS (Online & Batch) BE



openIMIS DB



Phase 1 : Locations

localhost/Locations.aspx

Home Inquiries and Policies Claims Administration Tools My profile Logout Search Insurance AD

2 Region(s)

| CODE | NAME |
|------|--------|
| R1 | Ulsin |
| R2 | Tahide |

3 District(s)

| CODE | NAME |
|------|-------|
| D1D1 | Ulsin |
| D1D2 | Hamam |
| D1D3 | Rapla |
| D1D4 | Liyed |

2 Municipality(s)

| CODE | NAME |
|------|----------|
| M1 | Assiouty |
| M2 | Remorogy |

3 Village(s)

| CODE | NAME | MALE | FEMALE | OTHER | FAMILY |
|------|--------|------|--------|-------|--------|
| V1 | Hosny | | | | |
| V2 | Ota | | | | |
| V3 | Rabwah | | | | |

Add Edit Delete Move Cancel

localhost/front/locations

openIMIS Search...

Locations

Claims

Health Facility Claims

Review

Batch Run

Administration

...

2 Region(s)

| CODE | NAME |
|------|--------|
| R1 | Ulsin |
| R2 | Tahide |

3 District(s)

| CODE | NAME |
|------|-------|
| D1D1 | Ulsin |
| D1D2 | Hamam |
| D1D3 | Rapla |
| D1D4 | Liyed |

2 Municipality(s)

| CODE | NAME |
|------|----------|
| M1 | Assiouty |
| M2 | Remorogy |

3 Village(s)

| CODE | NAME | MALE | FEMALE | OTHER | FAMILY |
|------|--------|------|--------|-------|--------|
| V1 | Hosny | | | | |
| V2 | Ota | | | | |
| V3 | Rabwah | | | | |

Add Edit Delete Move Cancel

Phase 1: all proxies

Deliverables:

- From current openIMIS

 - Login API

 - Pages without (head) menu

- From modularized openIMIS

 - Login

 - Coarse decomposition into “modules”

 - Main menu (by user profile?) ‘rebuilt’ (via contributions)

 - Assembly/deployment procedures (front only)

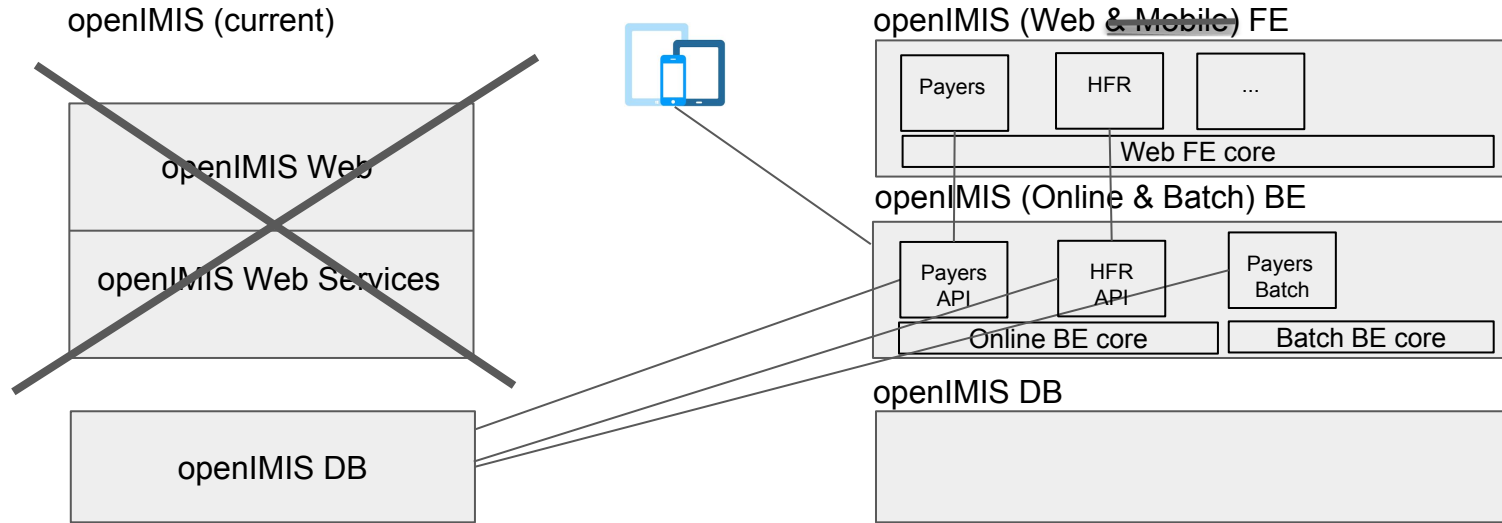
Risks:

- Wrong decomposition

- Opening security breach

Phase 2 : modularize (web) application

Migrating code only (no data migration) in **n iterations** (by module or group of modules)

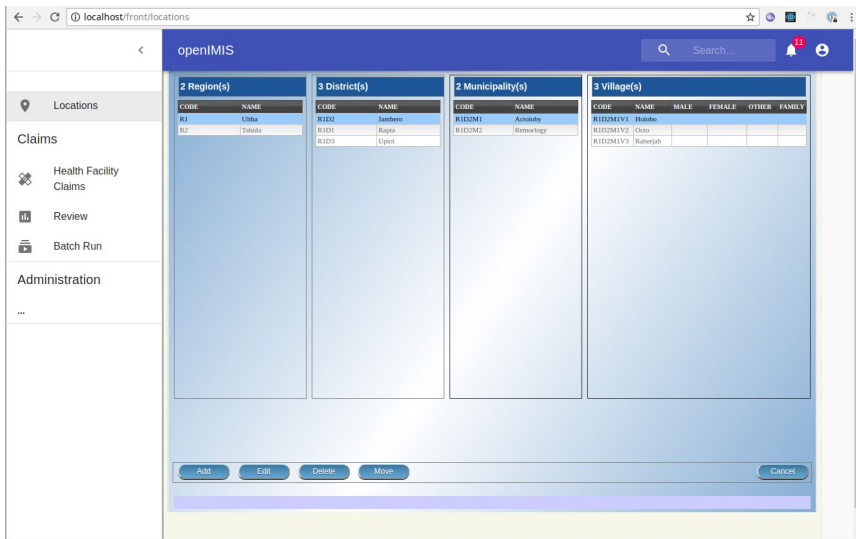


Phase 2: modularization iterations - backend

- django model from SQL-Server database: `django.db.models.*`
 - Enable django admin
- Wire REST API: `django rest framework`
 - and `django rest jsonapi`
 - and `django dynamic rest`
- Add REST 'actions'
 - with dispatched event
 - detaching from http request (start the broker,...)
- Implement batch processes
 - pagination & transaction
 - N+1 queries
 - ... and deploy in `airflow`

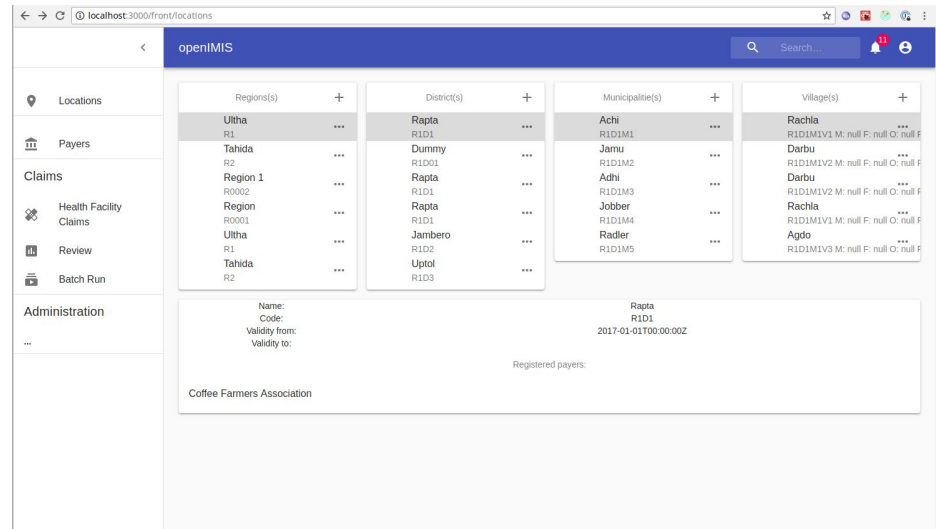
Phase 2: modularization iterations - frontend

- React components and contributions
- Redux for “dynamic” dependency



The screenshot shows the openIMIS application interface. The top navigation bar includes a search bar and a notification icon. The main content area is divided into four columns representing different levels of administrative divisions: Regions, Districts, Municipalities, and Villages. Each column contains a table with columns for CODE, NAME, and other attributes. The left sidebar contains navigation options: Locations, Claims, Health Facility Claims, Review, Batch Run, and Administration. At the bottom, there are buttons for 'Add', 'Edit', 'Delete', 'Move', and 'Cancel'.

| 2 Region(s) | | 3 District(s) | | 2 Municipality(s) | | 3 Village(s) | | | | | |
|-------------|--------|---------------|---------|-------------------|----------|--------------|----------|------|--------|-------|--------|
| CODE | NAME | CODE | NAME | CODE | NAME | CODE | NAME | MALE | FEMALE | OTHER | FAMILY |
| R1 | Ultha | R1D2 | Jamboro | R1D2M1 | Acsooby | R1D2M1V1 | Hokoko | | | | |
| R2 | Tahida | R1D3 | Rapta | R1D2M2 | Remerage | R1D2M1V2 | Uptol | | | | |
| | | R1D3 | Uptol | | | R1D2M1V3 | Rubersab | | | | |



The screenshot shows the openIMIS application interface with a detailed view of a location. The top navigation bar is the same as in the previous screenshot. The main content area is divided into four columns representing different levels of administrative divisions: Regions, Districts, Municipalities, and Villages. Each column contains a list of items with expandable options. The left sidebar is the same as in the previous screenshot. At the bottom, there is a section for 'Registered payers' with a list of names and codes.

| Regions(s) | District(s) | Municipalitie(s) | Village(s) |
|--|---|--|--|
| Ultha R1 Tahida R2 Region 1 R0002 Region R0001 Ultha R1 Tahida R2 | Rapta R1D1 Dummy R1D01 Rapta R1D1 Rapta R1D1 Jambero R1D2 Uptol R1D3 | Achi R1D1M1 Jamu R1D1M2 Adhi R1D1M3 Rachia R1D1M4 Radler R1D1M5 | Rachia R1D1M1V1 M: null F: null O: null F Darbu R1D1M1V2 M: null F: null O: null F Darbu R1D1M1V2 M: null F: null O: null F Rachia R1D1M1V3 M: null F: null O: null F Agdo R1D1M1V3 M: null F: null O: null F |

Name: Rapta
Code: R1D1
Validity from: 2017-01-01T00:00:00Z
Validity to:
Registered payers:
Coffee Farmers Association

Phase 2 : modularize (web) application

Deliverables:

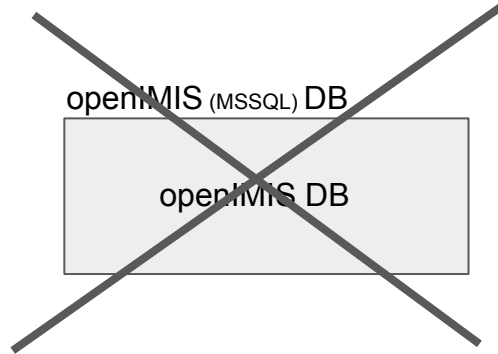
- .NET code & MS-SQL stored procedures taken over
- Mobile app connected to new platform

Risks:

- Database accessed from distinct app servers \Rightarrow concurrent changes (cache problem),...
- Performance in the (new) application layers and/or interactions with database

Phase 3 : database switch

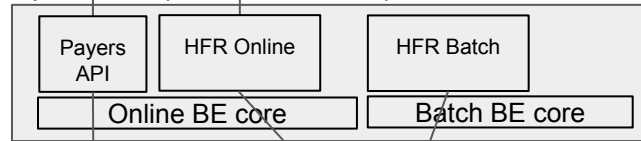
Migrating data only (no code migration) in **1 shot**, applying only data format transformation (datetime,...)
(no database structure change)



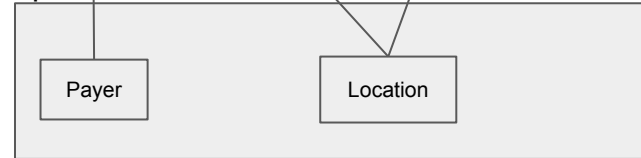
openMIS (Web & Mobile) FE



openMIS (Online & Batch) BE



openMIS (Postgres) DB



Phase 3: database switch

Deliverables:

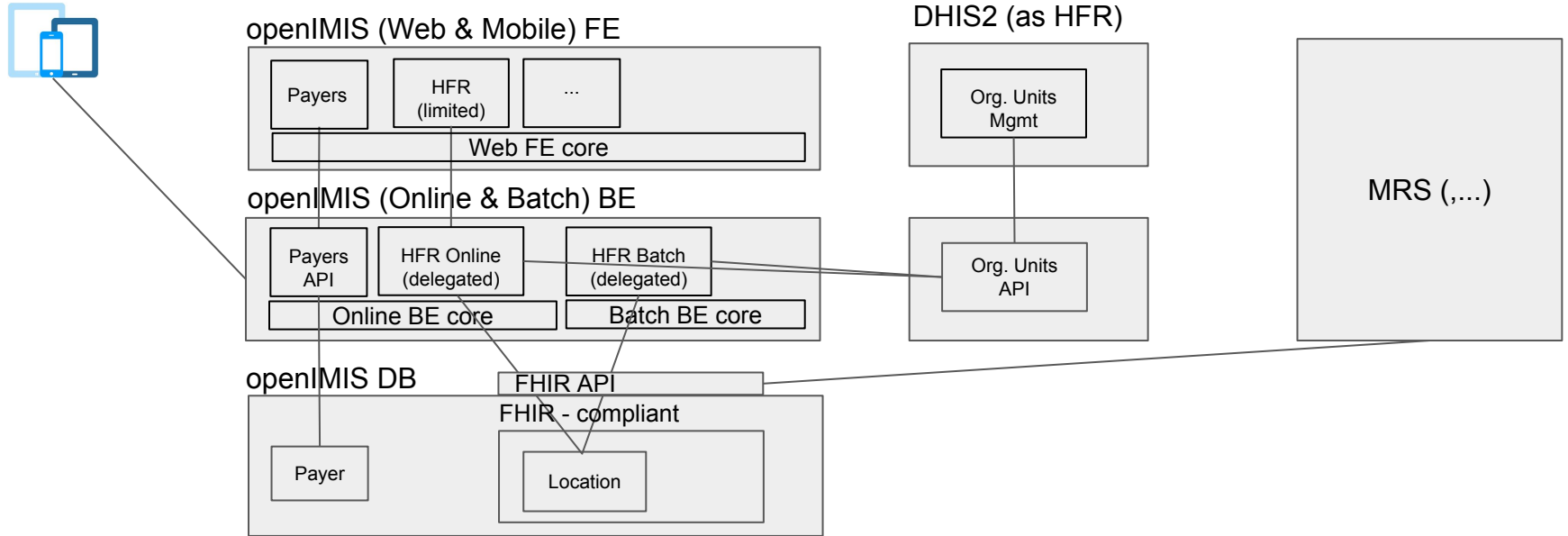
- Data migrated to the new database
- Application connected to new database

Risks:

- New platform stability (database tuning,...)
- Side effects of data formats changes (timestamps vs. datetime... with tz?)

Phase 4 : module refactorings

Refactoring (and adding) modules within **dedicated iterations**...



Postgres JSONB & Fhirbase

- Postgres JSONB

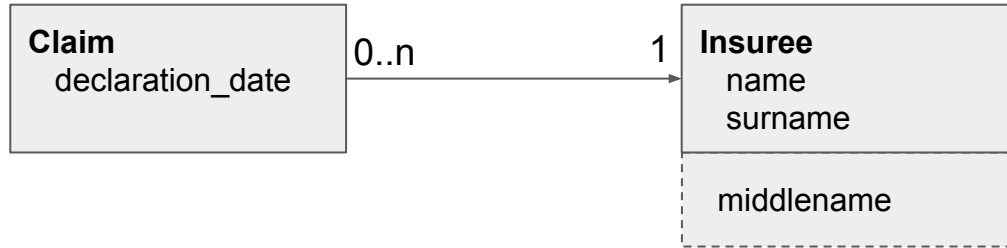
... where RDBS meets NoSQL

- FHIRBase

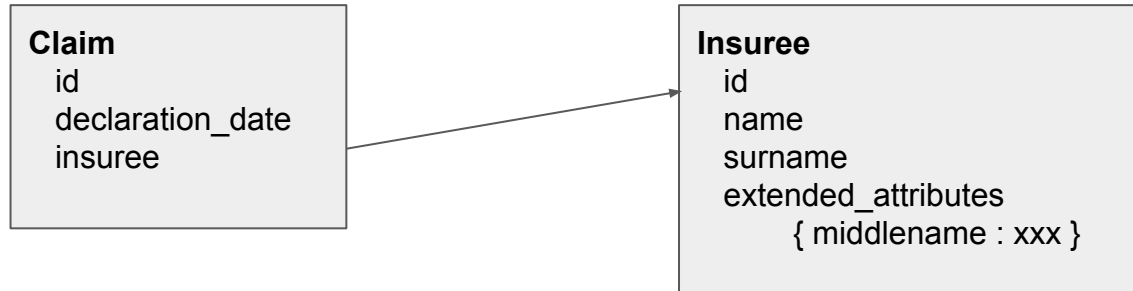


... the FHIR data model implemented in Postgres (using JSONB)

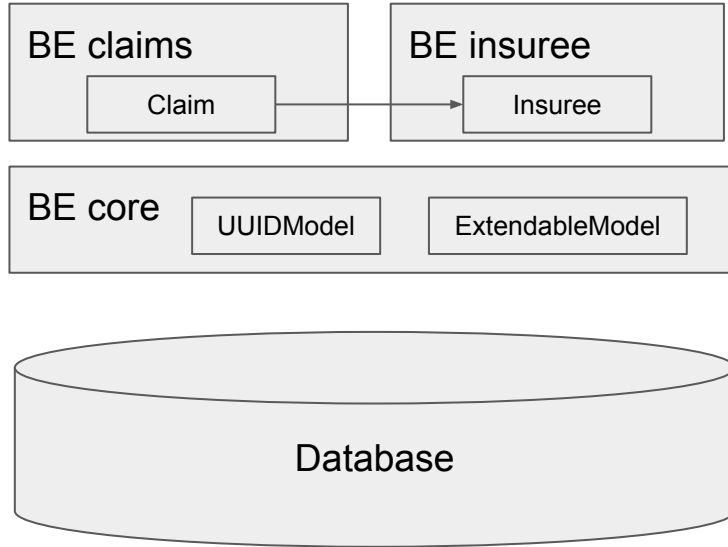
Modularity (contribution)



Database

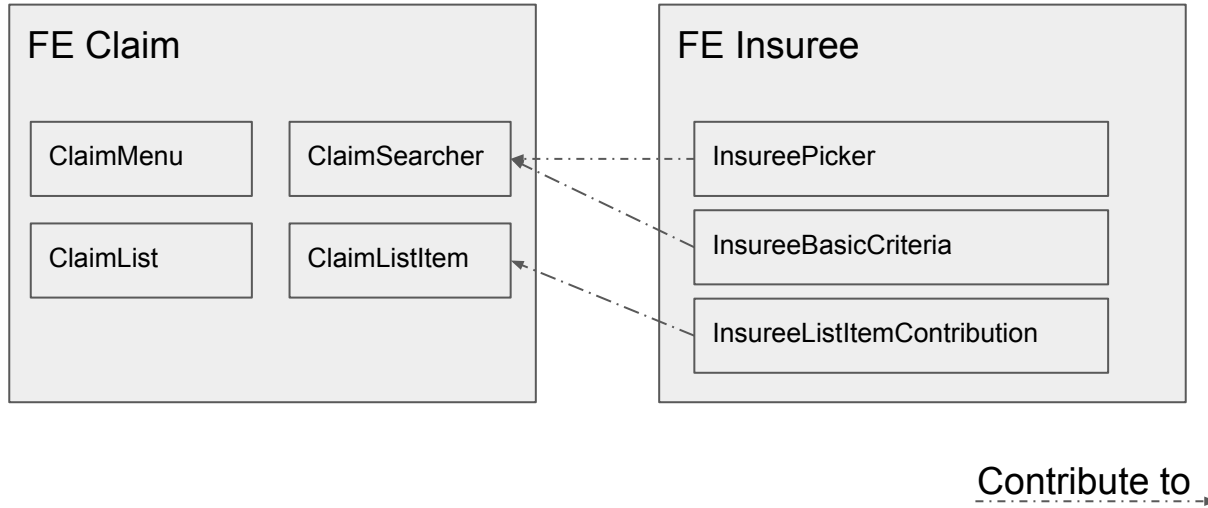


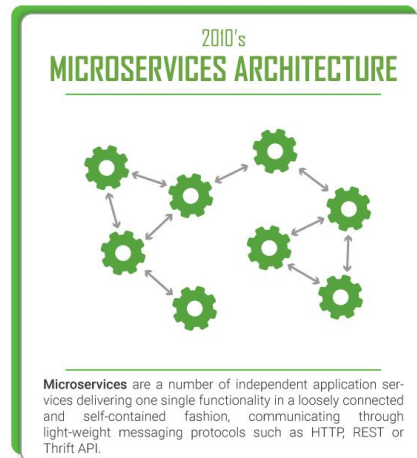
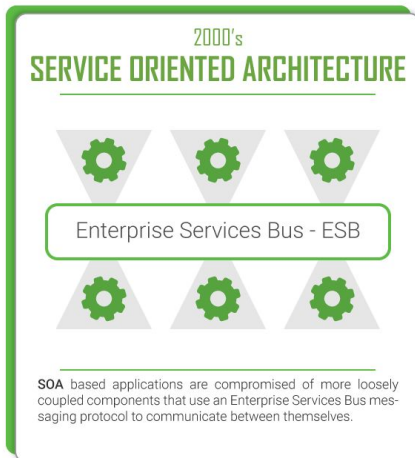
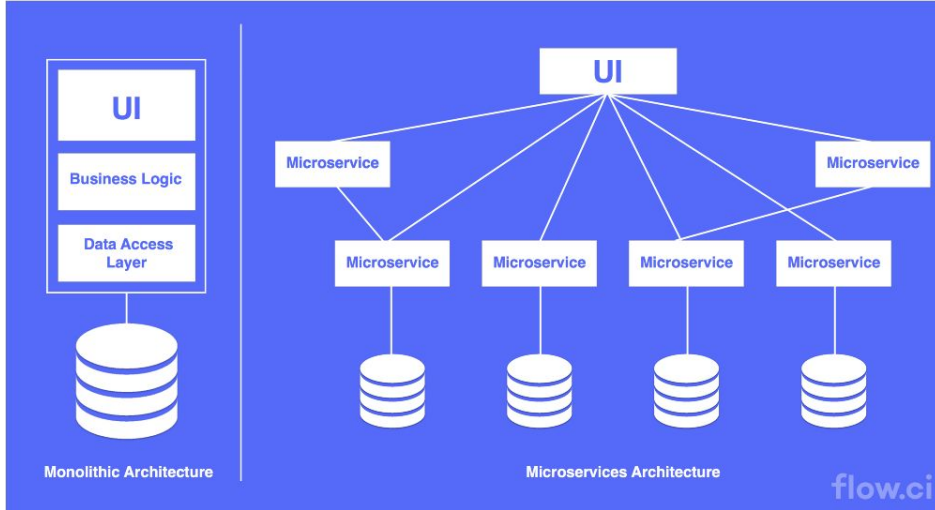
Modularity - Backend



API (Django REST Framework):
Side-relation vs. Embedded
Dynamic filtering

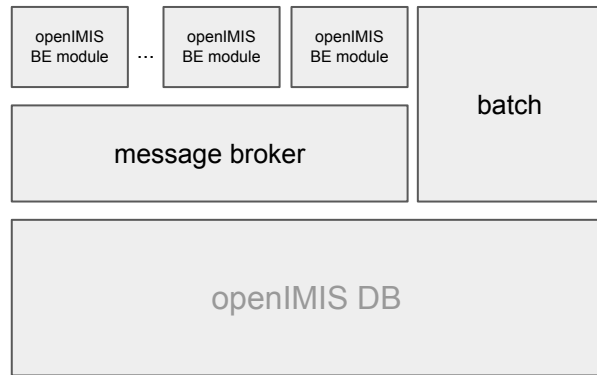
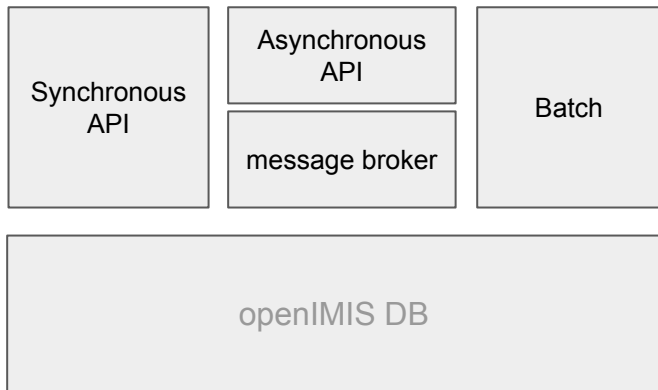
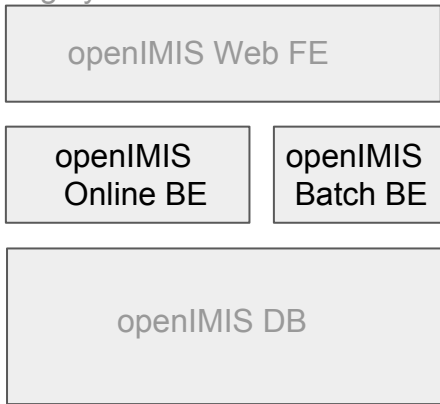
Modularity - Frontend





openIMIS BE: step beyond (too far?)

openIMIS -
Highly-distributed



Agenda (23/01)

Conceptual (cfr. wiki)

- About micro-services
- About NoSQL database

Migration strategy (“illustrated”)

Technical stack (showcase)

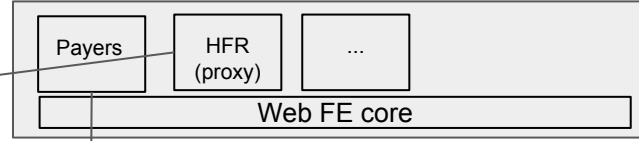
- Containers (docker)
- Front (Material-ui, React & Redux)
- Back (Django/python)
- *Database (SQL-Server > Postgres)*



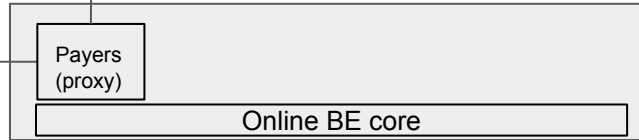
openIMIS (current)



openIMIS (Web & Mobile) FE



openIMIS (Online & Batch) BE



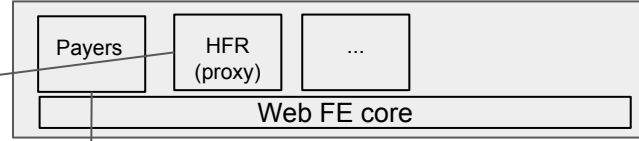
openIMIS DB



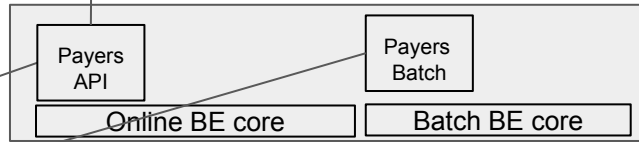
openIMIS (current)



openIMIS (Web & Mobile) FE



openIMIS (Online & Batch) BE



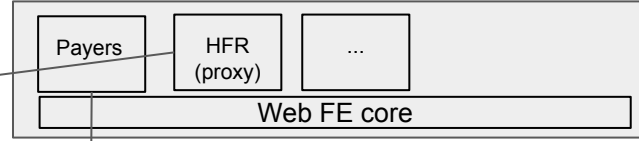
openIMIS DB



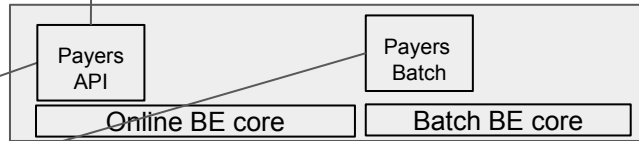
openIMIS (current)



openIMIS (Web & Mobile) FE



openIMIS (Online & Batch) BE



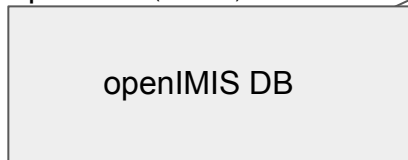
openIMIS DB



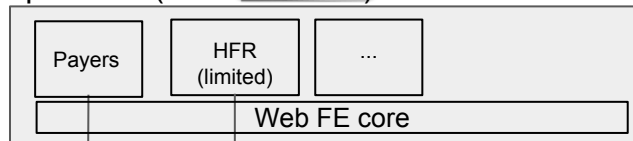
openIMIS (current)



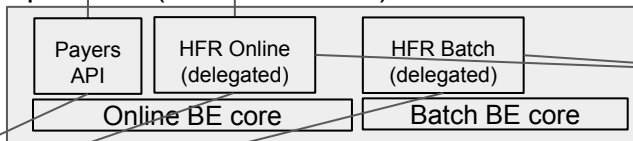
openIMIS (MSSQL) DB



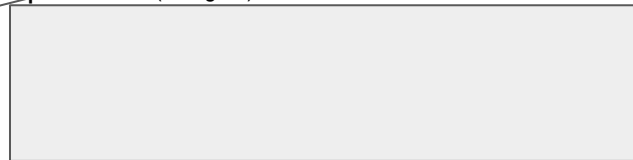
openIMIS (Web & Mobile) FE



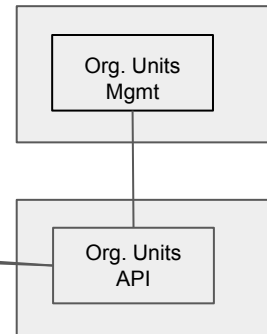
openIMIS (Online & Batch) BE

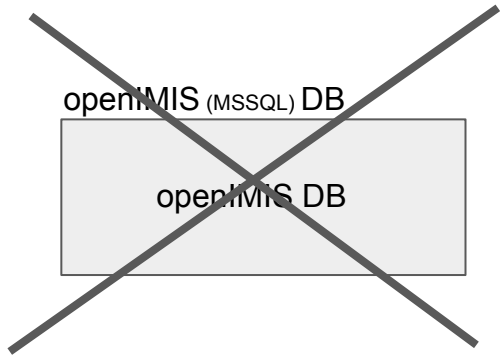


openIMIS (Postgres) DB

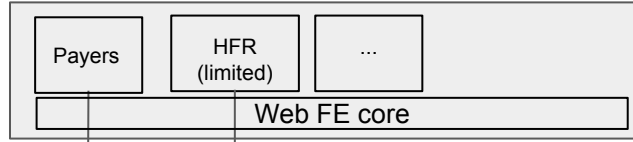


DHIS2 (as HFR)

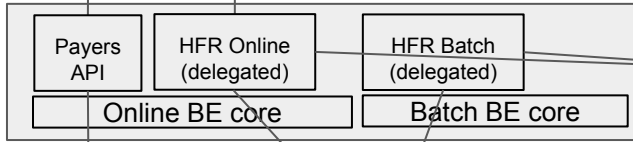




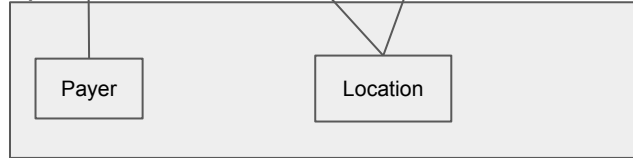
openMIS (Web & Mobile) FE



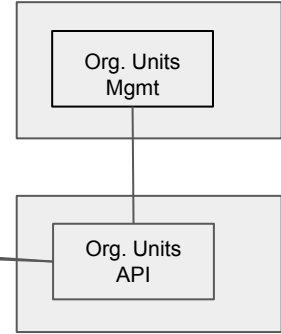
openMIS (Online & Batch) BE



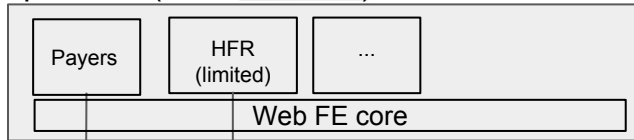
openMIS (Postgres) DB



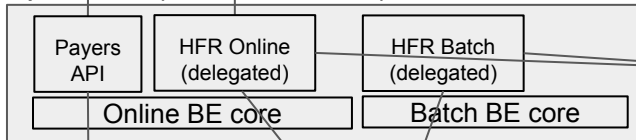
DHIS2 (as HFR)



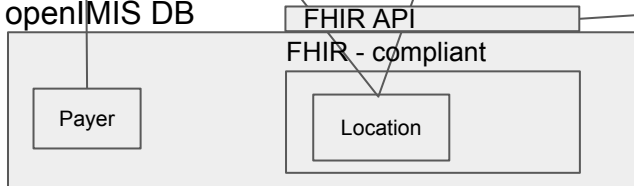
openIMIS (Web & Mobile) FE



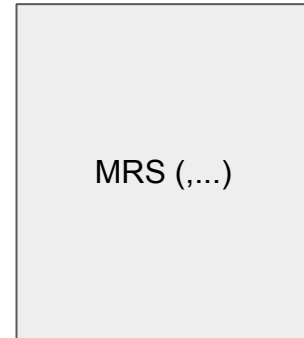
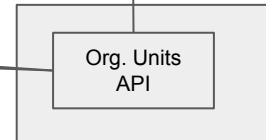
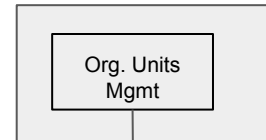
openIMIS (Online & Batch) BE



openIMIS DB



DHIS2 (as HFR)



Agenda 23/01

Conceptual (cfr. wiki)

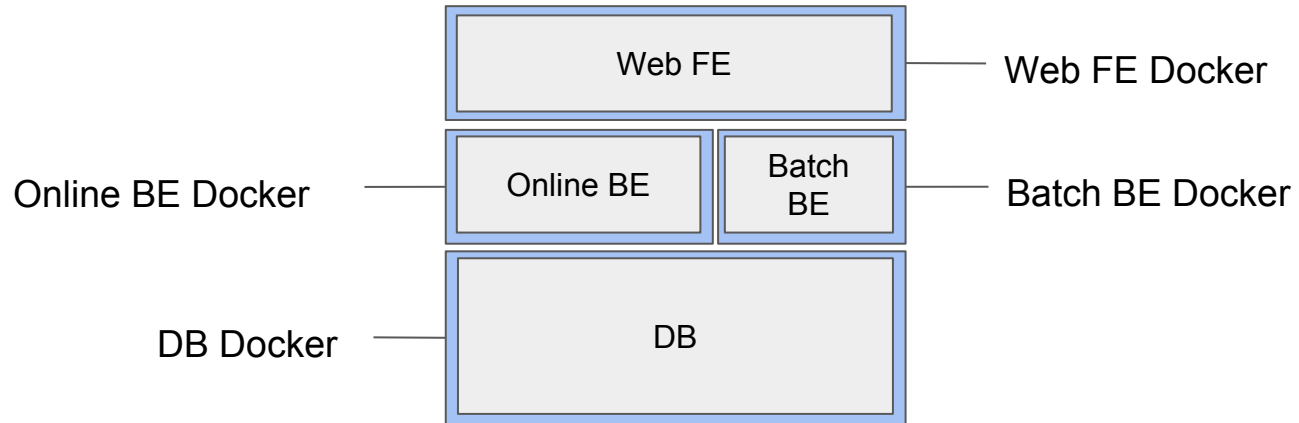
- About micro-services
- About NoSQL database

Migration strategy (“illustrated”)

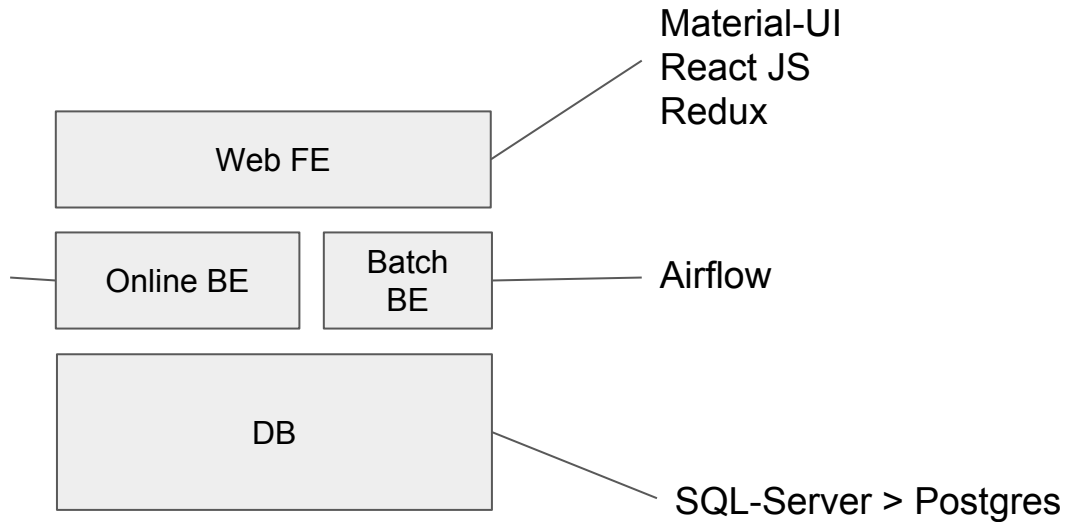
Technical stack (showcase)

- Containers (docker)
- Front (Material-ui, React & Redux)
- Back (Django/python)
- *Database (SQL-Server > Postgres)*





Django REST Framework
Django Admin
Django
Python



Agenda 31/01

FIHR integration

Feedback on Technology Stack (?)

Licensing

Modularity (contributions)



Licensing

docker: [Apache 2.0 license](#)

postgres: <https://www.postgresql.org/about/licence/> “a liberal Open Source license, similar to the BSD or MIT licenses.”

python: <https://docs.python.org/3/license.html> > <https://opensource.org/>

django: very light: <https://github.com/django/django/blob/master/LICENSE>

django REST Framework: <https://www.django-rest-framework.org/#license>

reactjs: MIT <https://reactjs.org/docs/how-to-contribute.html#license>

redux: MIT <https://github.com/reduxjs/redux/blob/master/LICENSE.md>

material-ui: MIT <https://material-ui.com/discover-more/backers/#sponsors-amp-backers>



