קבוצת ארכיטקטורה

FHIR® IL by 8400

על הקבוצה

משתתפים: ארכיטקטים, CTOs, מובילים טכנולוגים

מטרות הקבוצה

- היכרות עם פתרונות מומלצים (best practices) וארכיטקטורות אמיתיות מארגונים\מדינות אחרות
- גיבוש פתרונות משותפים ואינטראופרבילים
- התווית תקנים משותפים ומוסכמים (במקומות שלא מוגדרים היטב ב-FHIR)
- התמודדות עם אתגרים וסוגיות מורכבות – בעזרת חברי הקבוצה ויועצים בינלאומיים
- שיתוף של פתרונות אובדים ואתגרים

מהמפגשים הקודמים <<

אג'נדה

- Server vs Façade - פעם אחרונה ודי
- ניהול מזהים (וקשרים)
- REST vs Messaging
- אבלוס שרת FHIR בנתונים
- אנליטיקה ב-FHIR

תאום ציפיות

- זה דיון פתוח - תכוונו אותי לאזורים שמעניינים אתכם, תשאלו שאלות
- יש הרבה נושאים מעניינים - אם לא נספיק לעבור על הכל או אם תרצו התעמק - אתם מוזמנים לפנות אלינו גם אחר-כך

Server vs Façade

Façade

- **הרבה יותר קל/מהר להתחיל** - משתמש בכלים וידע קיימים
- **גמיש יותר ואינו דורש תכנון רב מראש** - מאפשר "לרמות" קצת ולסטות מ-FHIR בעת הצורך
- **אינו מצריך העברת נתונים יקרה ומורכבת**

- קל ליישם תרחיש בסיסי, אבל מסתבך מאוד מהר עם תרחישים קצת יותר מתקדמים (חיפוש, טרנזקציות, ניהול מזהים, היבטי ביצועים)
- קלות ה"רמאות" פוגעת בסופו של דבר בהטמעת התקן
- לא גנרי בכלל - כל תרחיש דורש יישום חדש

Server

- דורש השקעה (משמעותית) מראש (בחירה, רכישה, התקנה, תמיכה וכו')
- דורש תהליכים מסודרים לאבלוס וסנכרון נתונים
- דורש הבנה מעמיקה יותר של FHIR ותכנון מוקדם (ניהול מזהים, אסטרטגיית גישה, MDMוכו')

- מאלץ אתכם **להתיישר ל-FHIR** (וזה דבר טוב)
- **מספק הרבה פונקציונליות (מורכבת)** -out-of-the box (חיפוש, טרנזקציות, SMART וכו')
- **הרבה יותר גמיש וגנרי** (במינימום מאמץ)
- **ממשיך להתפתח יחד עם תקן**

Server vs Façade

Server

- דורש השקעה (משמעותית) מראש (בחירה, רכישה, התקנה, תמיכה וכו')
- דורש תהליכים מסודרים לאבלוס וסנכרון נתונים
- דורש הבנה מעמיקה יותר של FHIR ותכנון מוקדם (ניהול מזהים, אסט

- מאלץ אתכ
- **מספק הרבה** out-of-t
  box (חיפוש, טרנזקציות, SMART וכו')
- **הרבה יותר גמיש וגנרי** (במינימום מאמץ)
- **ממשיך להתפתח יחד עם תקן**

Façade

- **הרבה יותר קל/מהר להתחיל** - משתמש בכ
  קיימים
- **גמיש יותר ואינו דורש תכנון רב מראש** - מאפשר "לרמות" קצת ולסטות מ-FHIR בעת הצורך
- **אינ ומורכבת**

- ק תבך מאוד מהר
  ע (חיפוש,
  טרנזקציות, ניהול מזהים, היבטי ביצועים)
- קלות ה"רמאות" פוגעת בסופו של דבר בהטמ
  התקן
- לא גנרי בכלל - כל תרחיש דורש יישום חדש

לא חייבים לבחור אחד.
אפשרי לשלב את שניהם
בארכיטקטורה הארגונית

פתרונות מובנים לטווח ארוך

ציר הזמן / בשלות

פתרונות מהירים, קצרי טווח, ורטיקליים

אין דבר כזה
"façade" גנרי
(יש, אבל זה יקר מדי)

ניהול מזהים (וקשרים)

Identifiers

Logical ID

- “row id”

Business identifier

- T.Z.

References

Literal reference

- An absolute or relative URL with logical ID (or canonical)

Logical reference

- Reference to business identifier

Most of the reference-related APIs (partitions, chaining, includes, etc.) are not available (out of the box) with logical references

ניהול מזהים (וקשרים)

Example - Retrieve all Patients who have an Observation with code 567-8

With logical reference

Observation: Subject -> Patient(T.Z. 12345)
Patient: Identifier(T.Z.): 12345

Query 1:

GET [base]/Observation?code=567-8

Parse Observation object, extract identifiers, construct second query

Query 2:

GET [base]/Patient?identifier=http://acme.org/patient|12345,…

With literal reference

Observation: Subject -> Patient/abc
Patient: id:abc

Query 1:

GET [base]/Patient?_has:Observation:patient:code=567-8

Most of the reference-related APIs (partitions, chaining, includes, etc.) are not available (out of the box) with logical references

ניהול מזהים (וקשרים)

Why do I even need this?

- **When someone sends you Patient** – how do you avoid creating multiple copies of the resource?
- **Every time someone sends you Organization reference** – how do you know if you already have it and make sure the reference always points to the right resource?

Example: Master Data

Patient, Practitioner, Organization, etc.

Do you have full control of the clients? Can you trust them to do the right thing?

ניהול מזהים (וקשרים)

Some options for handling references on write:

- **Server internally resolves** logical reference to literal reference (customization)
  - Also consider messaging
- **Conditional reference** (out of the box)
  - According to HL7 – only in transactions
  - Most of servers support everywhere
- **Enforce proper** literal references
  - CapabilityStatement.rest.resource.referencePolicy

Handling references on read:

You can add custom logic for resolving logical references, but it'll probably be too complicated for anything, but the basic scenarios and you'll probably have to break away from the standard

Do you have full control of the clients? Can you trust them to do the right thing?

Rest vs Messaging

There are other paradigms, but we won't touch them here

And in general – look at "Approaches to Exchanging FHIR Data"

More info>>

Rest vs Messaging

REST

- “Classic” FHIR – most of the tools & platforms target it and support it well
- SMART
- Loosely coupled - more interoperable out of the box
- Easier for read, gets complicated fast for write
- How much control do you have over your clients?

Messaging

- Older HL7v2 approach
- But also aligns well with modern architectural practices
- Tightly coupled - less interoperable out of the box (but can be decoupled and very scalable at transport level)
- REST messages – built-in operation - *$process-message*
- You are kind-of already using it (בספות)

Rest vs Messaging

You can combine the two – MoH scenario - CQRS

- Most of the data comes from outside
- Significant diversity in data providers (clients) capabilities
- Complicated intake pipeline (Validation & DQA, Identifiers mapping, MDM, etc.)

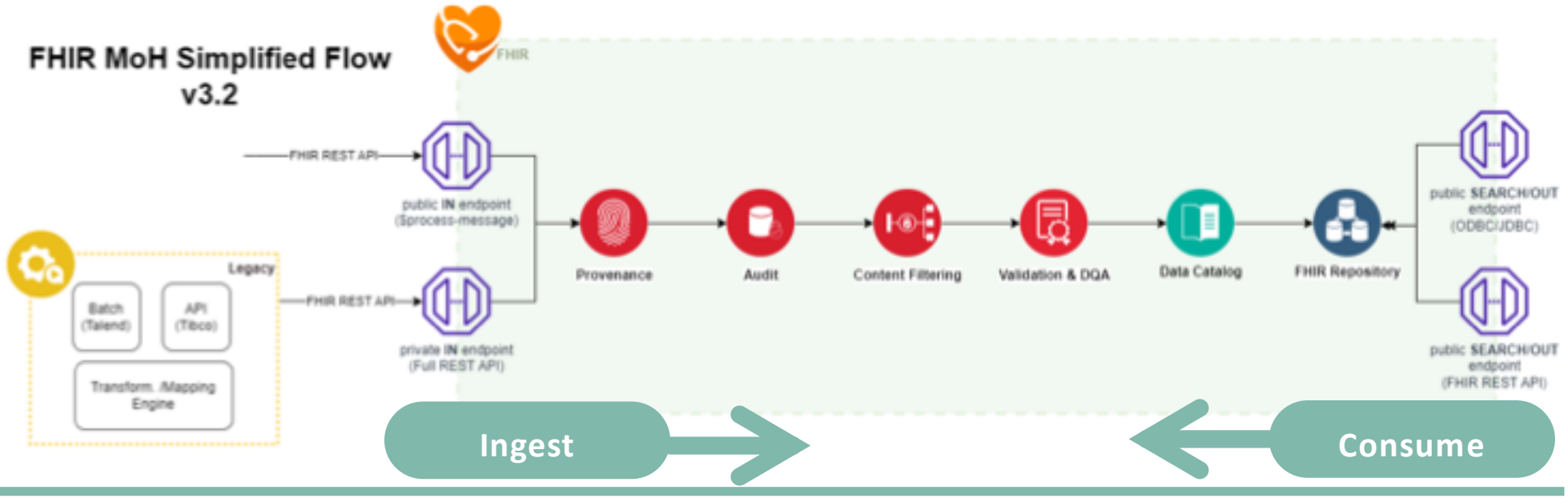
FHIR IL by 8400

More info>>

Rest vs Messaging

You can combine the two – MoH scenario - CQRS

- Most of the data comes from outside
- Significant diversity in data providers (clients) capabilities
- Complicated intake pipeline (Validation & DQA, Identifiers mapping, MDM, etc.)



More info>>

Questions | אבלוס שרת FHIR בנתונים

- Where is the data coming from?
  - Internal or External systems?
  - Services (i.e. – B2B) or end-user clients (i.e. – B2C)?
  - Do you have one source system or multiple
- Do you need to feed data back to source (i.e. – bidirectional sync)?
- What are the “freshness” requirements? Uniform for all data or variable?
- Do you have history in your source system(s)? Do you want it in FHIR?
- Do you have stable identifiers in source system for everything? Are you sure? (consider case of splitting source record into multiple FHIR resources)
- Will you load business identifiers or logical IDs?
- Do you need to propagate permissions from the source system?
- What about regulatory requirements – can you handle all data equally/land in the same FHIR repository?
- Do you need terminology mapping when you load the data into FHIR?
- What about volumes? How much data are you loading? Is it big initial load and then small syncs? Projected peak loads?
- Do you need to support reflow scenario?

Example | אבלוס שרת FHIR בנתונים

Assumptions

- Load data from org. EMR & ADT systems into FHIR server
- Read-only copy (so no bidirectional sync)
- Limited dataset (Patients, Observations for lab results, Procedures)
- Initial bulk load with subsequent hourly updates (for reflow drop repository and start initial load again)
- ~100 mil resources load + ~1000 resources hourly
- No history
- All data goes to the same FHIR repository
- No permissions propagation from source
- Add SNOMED code for Observations and Procedures during load
- Resources without stable business identifiers will have those generated using deterministic logic

Process

Initial Load

1. Use you favorite ETL tool to extract data from the source system, convert it into FHIR resources wrapped in a transaction bundle with conditional references, add terminology mapping (use caching here) and save into files/db as JSON blobs segmented by resource type
2. Disable access to the FHIR server for everyone but your ETL
3. Wipe target FHIR repository
4. Use your favorite ETL tool to send prepared JSON bundles to FHIR server
   - Mind references – send “referred” resources (i.e. – Patient) first and “referring” resources (i.e. – Observation) second
   - Within single resource type use multiple threads – FHIR servers can handle concurrency
5. Re-enable access to FHIR server

Ongoing sync

1. Start an hourly scheduled task on your favorite ETL to extract delta from the source systems, generate transaction bundles using the same logic as above and send them to FHIR server.
   - Make sure to mind references and resource order here as well

Example | אבלוס שרת FHIR בנתונים

Assumptio

- Load data from org. EM
  FHIR server
- Read-only copy (so no b
- Limited dataset (Patien
  results, Procedures)
- Initial bulk load with su
  (for reflow drop reposit
  again)
- ~100 mil resources load
  hourly
- No history
- All data goes to the sam
- No permissions propag
- Add SNOMED code for
  Procedures during load
- Resources without stab
  will have those generate
  logic

Bulk import

- Most vendors support bulk import (Intersystems, Firely, Hapi/SmileCDR, Azure, etc.)
- Some try to adhere to Bulk Data Import IG (https://github.com/smart-on-fhir/bulk-import/blob/master/import-manifest.md), some do their own custom thing
- Most use ndjson
- Many support updates too
- Keep in mind that many skip a lot of steps (like validation) and load directly into DB
- Most allow you to provide your own logical IDs

Not sure you needed it – most servers can ingest **10s to 100s of millions** of resources per day on reasonable HW via standard API

nvert it into
ferences, add
ON blobs

erver
and

can handle

from the
as above and

il

אנליטיקה ב-FHIR

Might not be your use-case (if most of your data doesn't originate in FHIR). Still might be worth thinking about for the long term – as more and more data will be "born" in FHIR

Depends on type of analytics

- Native FHIR format is good for big data / machine learning (and Bulk FHIR NDJSON is even better)
- Native FHIR is also good for transformers, however:
  - FHIR tends to be verbose, so mind your token limitation. One (modest) resource at a time. Another option is preprocessing/custom tokenizers
  - No existing foundation models (AFAIK) – so additional training might be needed
- Native FHIR format is pretty bad for "classic relational" analytics (very hierarchical, denormalized, complicated data types, etc.)
  - In other words – you can ask your analyst to write JSON_VALUE/JSON_QUERY select over native FHIR data, but there is a good chance heavy objects will be thrown at you

אנליטיקה ב FHIR

So, how do you do “classic relational” analytics on FHIR? You map it to relational model

How to map?

Some FHIR servers (e.g. - Intersystems) provide a way to “project” FHIR data into relational representation without physically moving the data

- **Pros:** you don’t need to move the data and you can create as many virtual views as you like; fewer moving parts
- **Cons:** strong coupling of op and analytical systems; query performance might be an issue

Other FHIR servers provide a way to stream the data out with minimal latency and map it into relational representation on the way (and you can build your custom solution as well – not too hard)

- **Pros:** decoupling of op and analytical systems; independent scale
- **Cons:** You need to physically move the data and make sure it doesn’t “leak”

Map into what?

More standard:

- SQL on FHIR (BigQuery uses it – and there is automatic export from GCP Cloud Healthcare FHIR server)
- OMOP (tailored for analytics and there are some tools to convert FHIR into OMOP, but it’s very opinionated, so might not fit your needs/be difficult to convert, probably overkill for most unless you are already using it)

More niche:

- Fhirbase

Your own:

- If you already have standard organizational data model

שאלות?