Build a documented and tested healthcare API in 30 minutes

U.S. DIGITAL SERVICE // SHELBY SWITZER // NOV 2020

Who am I?

- Shelby Switzer (they/them)
- Twitter: @switzerly
- Github: switzersc, switzersc-usds
- US Digital Service @ the Department of Health and Human Services (HHS): usds.gov // @usds
- Civic tech, healthcare interoperability, APIs
- Civic tech blog: civicunrest.com

Agenda

- Learning goals (a.k.a. What's in it for you)
- Intro to the concepts
- Let's go!
  - Dependencies
  - Steps
- Tips
- Where to from here?
- Resources

Learning goals

(a.k.a. What’s in it for you)

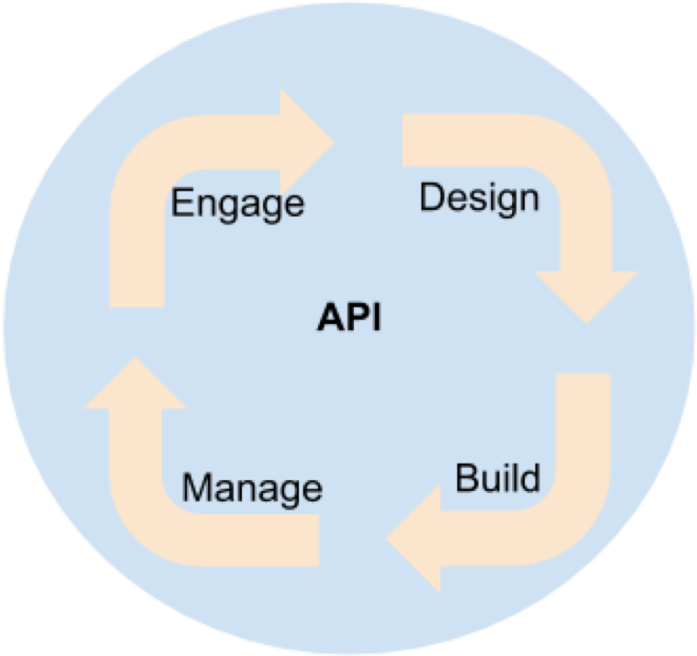
What's in it for you

- Discover new open source tools
- Get hands-on with FHIR and the OpenAPI Specification
- Practice a design-first API development paradigm:
  - Design -> Mock -> Iterate -> Develop

Intro to the concepts

API concepts

API Product Lifecycle



OpenAPI Specification

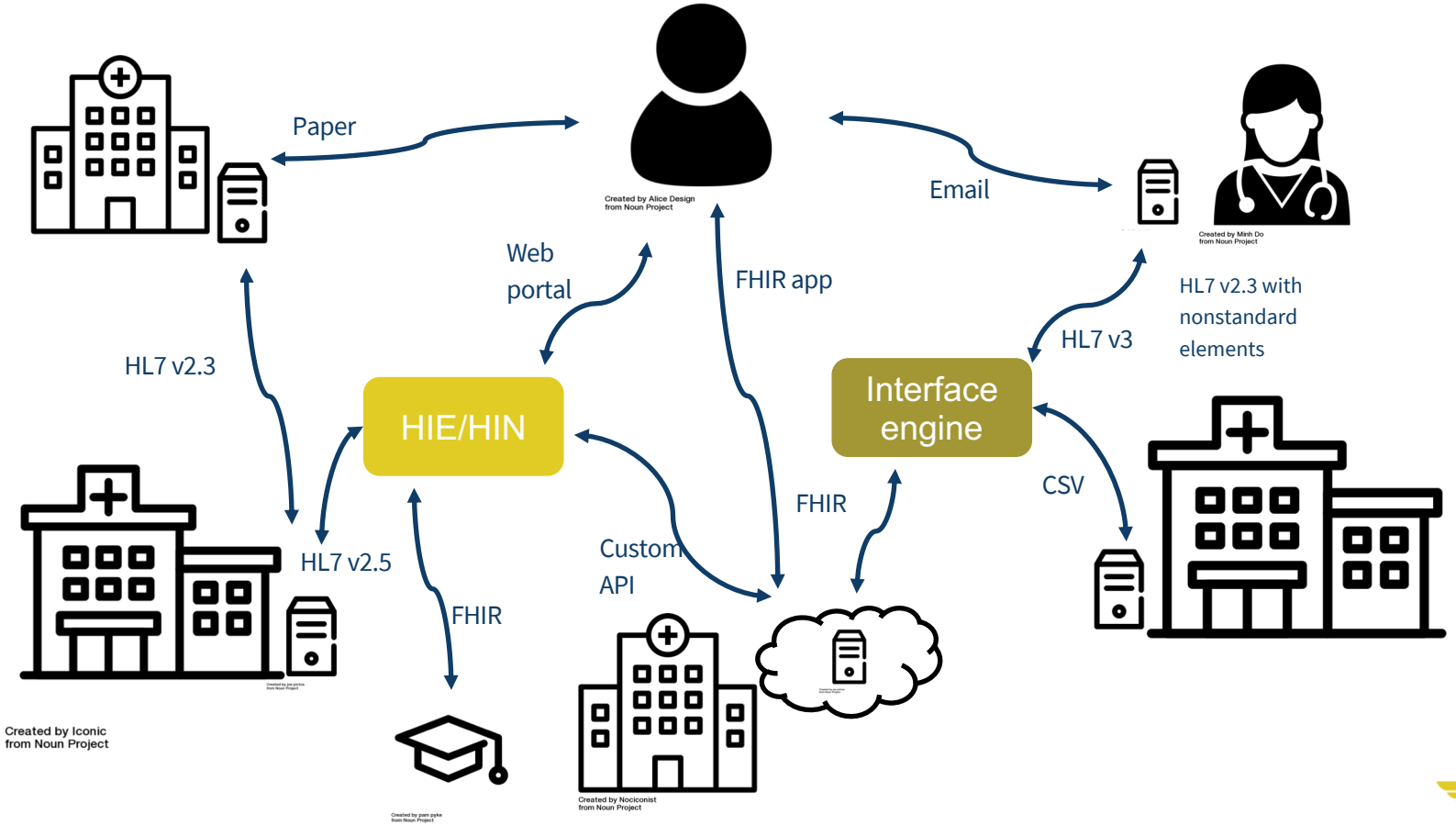
- Machine-readable API definition format
- Acts as the API contract
- Can act as the backbone for API lifecycle
- Formerly known as Swagger 

Healthcare concepts

Healthcare APIs

- Customization over standardization (workflow, business, tech)
- Many on-prem legacy systems
- Hugely variable tech, though now aspiring to “interoperability”
- Policy and government are major drivers of interoperability

Healthcare integration ecosystem (2000-today)



4 LEVELS OF HEALTHCARE INTEROPERABILITY

Foundational

Basic requirements to securely communicate data between systems

Structural

Format, syntax, and organization of data exchange

Semantic

Common underlying models and codification of the data, with shared definitions and meaning to user

Organizational

Includes governance, policy, social, legal and organizational considerations

Source: HIMSS, https://www.himss.org/what-interoperability

FHIR

- Fast Healthcare Interoperability Resources
- More modern replacement of HL7
- REST-aspiring
- Still early in adoption
- Designed for machines, not humans
- Profiles and capability statements act as API definition
- Implementation Guide (IG) is standard documentation
- Not designed to be compatible with OpenAPI

Let’s go!

Dependencies

Dependencies

General	Today
Machine-readable API definition	OpenAPI Specification
Server generator	OpenAPI Enforcer Middleware*
Documentation generator	ReDoc**
Testing library	OpenAPI Chai Validators

* NodeJS
** Internet browser

Steps

1. Start with the user & problem
2. Create your FHIR-compliant OpenAPI definition
3. Generate human-friendly documentation
4. Generate mock server
5. Build functionality
6. Test

1. Start with the user & problem

I’m a doctor with my own practice and I’d like to manage a list of patients who are eligible for immunizations. I have several partners at other clinics and in public health, and we all want to work from the same list to divide and conquer outreach to these eligible folks in our community and track who has received their immunizations.

2. Create FHIR-compliant OpenAPI definition

Design, then iterate

Which FHIR models to use? Let’s start with Patient and iterate.

Is Patient enough? Is JSON enough or do we need to add XML for our users?

3. Generate human-friendly documentation

Documentation options

- Lots of free SaaS options
- Lots of open source options we can integrate into our code and CI/CD pipeline

We’re just getting started, so let’s use a free option for now: ReDoc.

(ReDoc is open source so we can build on it later)

4. Generate mock server

Mocking the server with OpenAPI Enforcer

- Generates Express JS middleware from OpenAPI definition
- Auto-populates responses with fake values
- Uses examples from the OpenAPI definition if provided
- Extensible: You can build actual functionality once you're ready to move past mocks

5. Build functionality

Building the server with OpenAPI Enforcer

- Pick and configure your data store
- Add a `controllers` directory and a controller file per resource
- Add x- options to OpenAPI definition

6. Test

Testing options

- Automated code testing: open source libraries
- Can integrate into CI/CD
- Dev and production testing: Postman, Paw, Runscope, etc

Tips

Tips & Gotchas

- The full OpenAPI definition for FHIR core capabilities is huge and contains numerous web references that slow down tooling.
- JS not your thing? Many other languages have tooling support.
- You can get far with just a mock.
- Add examples to your OpenAPI definition to have a more realistic mock.

Where to from here?

Where to from here?

- Build more FHIR APIs!
- Add to the open source tooling ecosystem supporting FHIR, OpenAPI, and the broader API lifecycle
- Work towards greater harmony between FHIR and industry-agnostic API standards, tools, and best practices

Resources

Resources

- https://www.openapis.org/
- https://openapi.tools/
- https://byu-oit.github.io/openapi-enforcer-middleware/
- https://redocly.github.io/redoc/
- https://github.com/Redocly/redoc
- https://github.com/openapi-library/OpenAPIValidators
- https://build.fhir.org/ig/HL7/US-Core-R4/CapabilityStatement-us-core-server.html
- https://github.com/switzersc-usds/healthcare-api


U.S. DIGITAL SERVICE
Thank you!
USDS.gov/apply