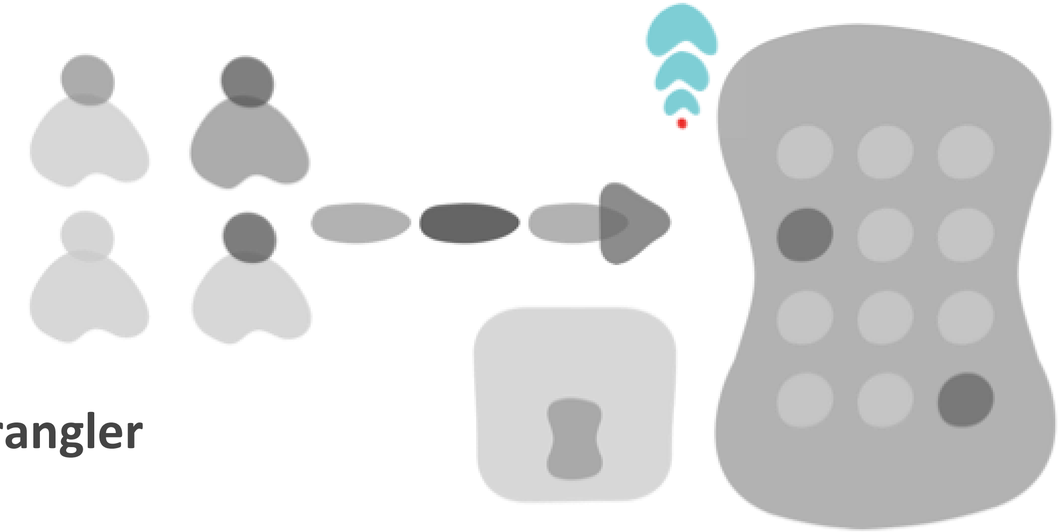
Architecture Katas

Autumn 2024: DCC



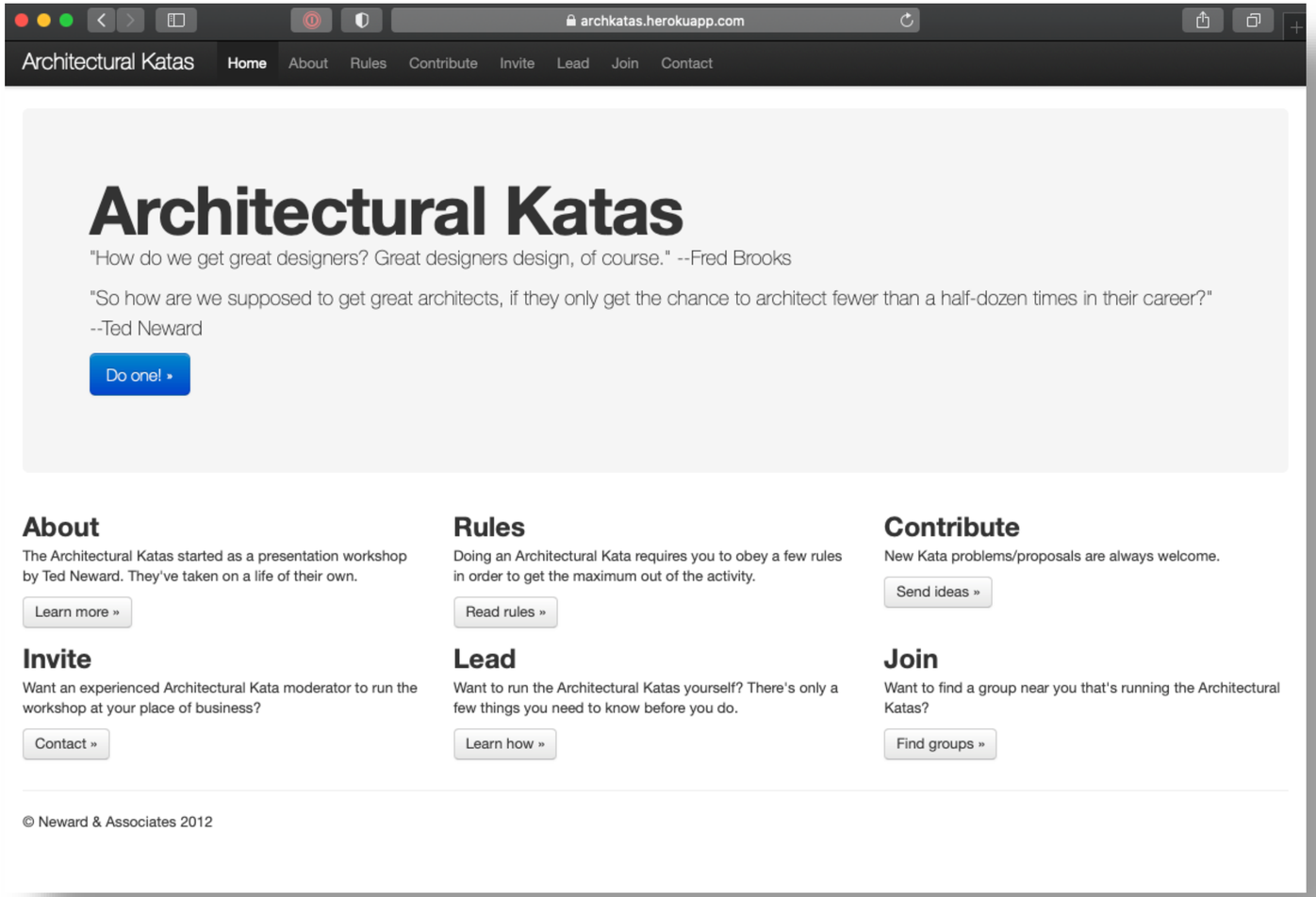
Neal Ford
Thoughtworks
Director / Software Architect / Meme Wrangler
https://www.nealford.com

Mark Richards
Independent Consultant
Hands-on Software Architect, Published Author
Founder, DeveloperToArchitect.com
@markrichardssa

Contest Kickoff

Introduction

Where did this idea come from?



archkatas.herokuapp.com

Architectural Katas | Home | About | Rules | Contribute | Invite | Lead | Join | Contact

Architectural Katas

"How do we get great designers? Great designers design, of course." --Fred Brooks

"So how are we supposed to get great architects, if they only get the chance to architect fewer than a half-dozen times in their career?" --Ted Neward

Do one! »

About

The Architectural Katas started as a presentation workshop by Ted Neward. They've taken on a life of their own.

Learn more »

Rules

Doing an Architectural Kata requires you to obey a few rules in order to get the maximum out of the activity.

Read rules »

Contribute

New Kata problems/proposals are always welcome.

Send ideas »

Invite

Want an experienced Architectural Kata moderator to run the workshop at your place of business?

Lead

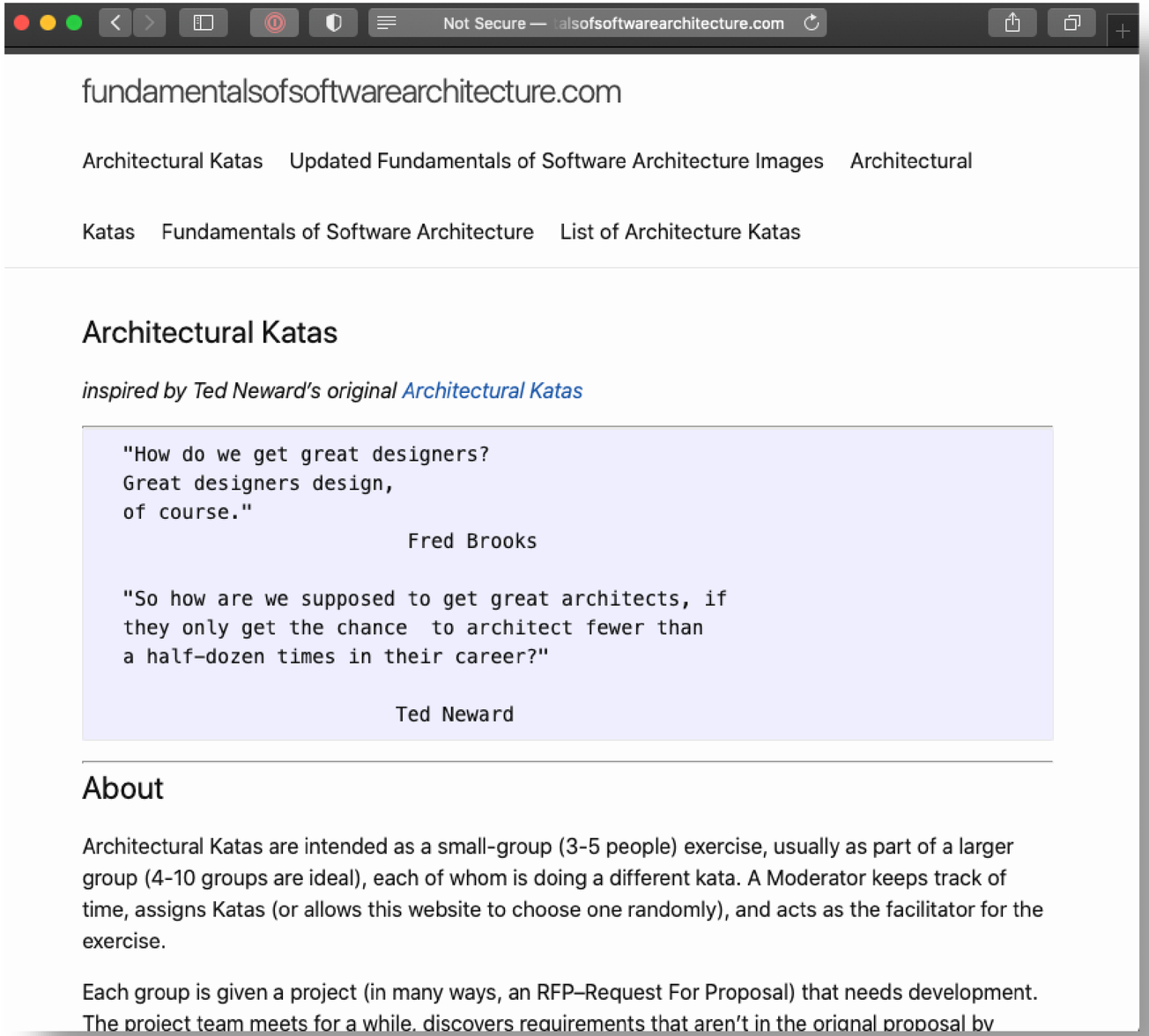
Want to run the Architectural Katas yourself? There's only a few things you need to know before you do.

Join

Want to find a group near you that's running the Architectural Katas?

...and then...

http://fundamentalsofsoftwarearchitecture.com/katas/




...and then...

LIVE ONLINE TRAINING

Architectural Katas

Topic: Software Development

Late 2020…

NEAL FORD

October 20, November 17 & December 3, 2020

10:00am – 12:00pm EST

This course has ended.

What you'll learn Instructor Schedule

New information after 10/20 kickoff:

Meet the Judges

Sarah Wells

Sarah Wells is a technology leader, consultant, and conference speaker with a focus on microservices, engineering enablement, observability, and DevOps. She has over 20 years of experience as a developer, principal engineer, and tech director across product, platform, SRE, and DevOps teams. She spent over a decade working at the Financial Times as it transitioned from 12 releases a year to more than 20,000 and adopted the cloud, microservices, and DevOps.

David Bock

David Bock is the vice president of strategic development at Core4ce, where he helps turn new ideas into successfully executed business plans. Previously, David was the VP of tech and engineering mission support at Decisiv, where he was responsible for internal IT operations, site reliability engineering, quality assurance, security, customer service, and the company’s release and triage teams. David served as the editor of O'Reilly's OnJava.com website, has been published in several books and magazines, and frequently speaks on technology and team processes at software conferences.

Chelsea Troy

Chelsea leads the machine learning operations team at Mozilla. She also teaches in the Master’s Program in Computer Science at the University of Chicago. Her online workshop, *Fundamentals of Technical Debt*, is available On Demand through the O’Reilly platform, and she also gives live courses about machine learning, large language models, and product thinking.

Meet the SME

Odie Gray

Odie Gray is a disabled military veteran, cybersecurity professional, and businessman of 17+ years specializing in business strategy development and transformation. As a thought leader in this space, he’s delivered strategic value to his clients by developing new cybersecurity service offerings and programs and managing teams for industry-leading organizations.

The Process

https://www.developertoarchitect.com/downloads/worksheets.html

https://www.developertoarchitect.com/lessons/lesson112.html

Architecture Characteristics Worksheet

System/Project: _____

Architect/Team: _____ **Date:** _____

Candidate Architecture Characteristics

performance	data integrity	deployability
responsiveness	data consistency	testability
availability	adaptability	abstraction
fault tolerance	extensibility	workflow
scalability	interoperability	configurability
elasticity	concurrency	recoverability

others: _____

_____

_____

a / b denotes characteristics that are related; some systems only need one of these, other systems may need both

Top 3	Driving Characteristics
☐	1.
☐	2.
☐	3.
☐	4.
☐	5.
☐	6.
☐	7.

Instructions

- Identify no more than 7 driving characteristics.
- Pick the top 3 characteristics (in any order).
- Implicit characteristics can become driving characteristics if they are deemed *structural* concerns.
- Add additional characteristics identified that weren't deemed as important as the list of 7 to the *Others Considered* list.

Implicit Characteristics

feasibility (cost/time)

security

maintainability

simplicity

Others Considered

_____

_____

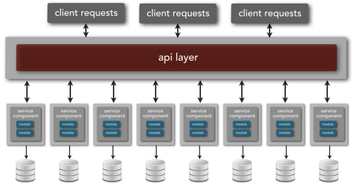
_____

_____

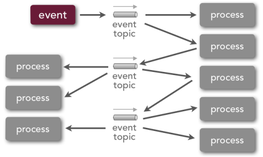
Created by Mark Richards, DeveloperToArchitect.com

Your Architectural Kata is...

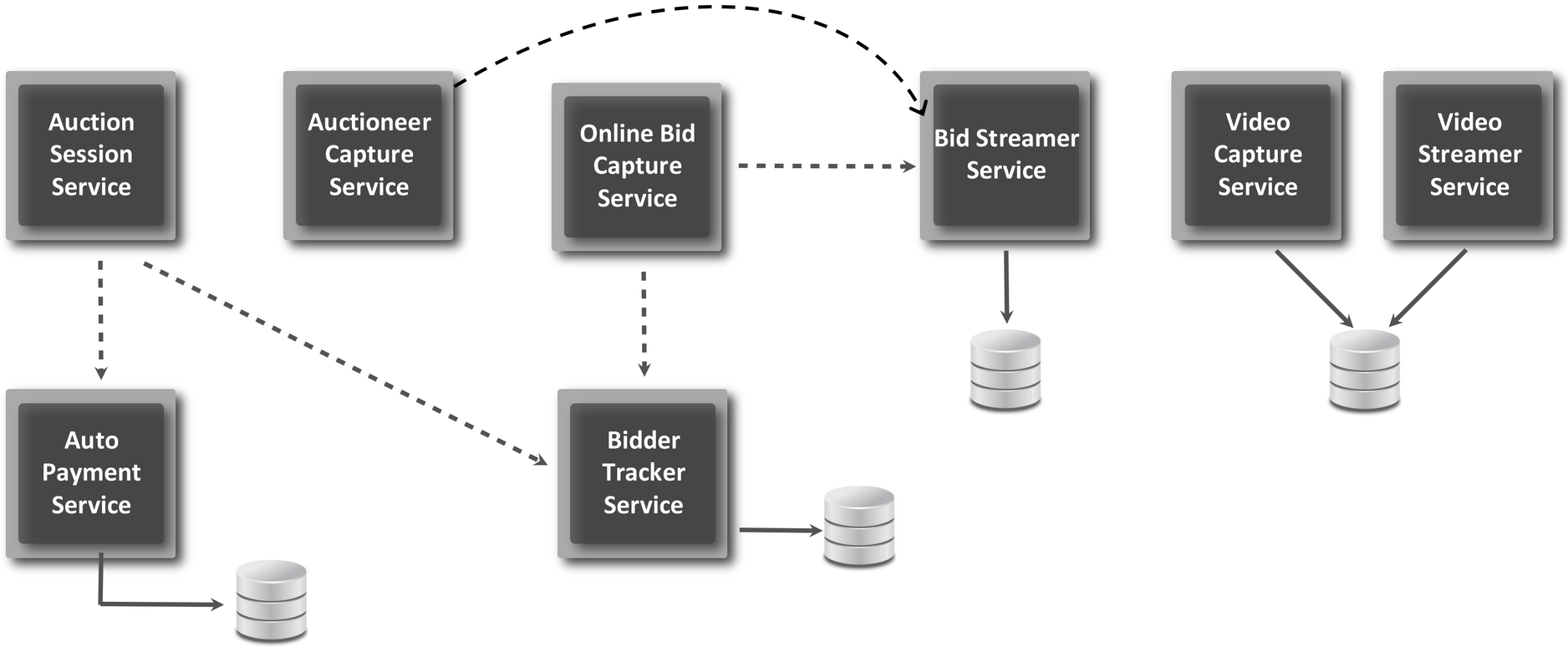
Going Going Gone!



microservices architecture



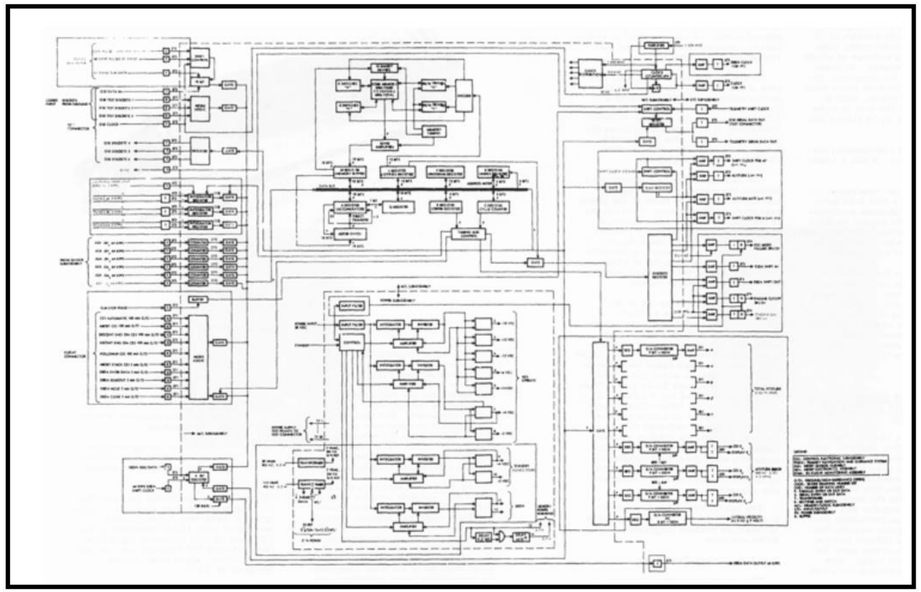
event-driven architecture



architecture decision records

ADR 001: Use the microservice architecture style with containerization

Farmacy Food is a start up company and does not have a sizeable team of experienced developers available. The overarching architecture style for the Farmacy Food system should be simple, easy to create, maintain and **evolve**. Finding developers that can create and evolve the system, as well as tools and frameworks that support the system should not require heaps of money. In other words, Farmacy Food is not in a position to be an *early adopter*, and should hence adopt an established architecture style that supports evolution.

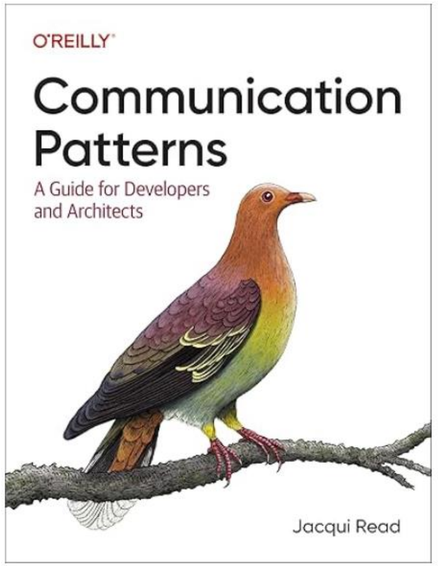


The Katalog

https://github.com/thekatalog

READ THE ARCHITECTURE

KATALOG



www.oreilly.com/library/view/communication-patterns/9781098140533/

Judges Criteria

Clarity of narrative, organization, and supporting documentation

Structure

- Problem Background
  - Business Goals Drivers
  - Requirements
  - Considerations & Data Criticality
  - Actors & Actions
  - RAID Log
  - Analyses Of 3rd Party Tools
  - Glossary
- Solution Background
  - Vision
  - Architecture Principles
  - Proposed Scenarios
- ARDs
  - 000. Use ADR
  - 001. ADR Template
  - 002. Build The System Using Iterations
  - 003. Camera Update Strategy
  - 004. Type Of User Notification
  - 005. Choose The First Labelling Platform Integrat
  - 006. Choose The First Model Training Integration
  - 007. Choosing The Microservice Architecture
  - 008. Merge Authentication And Authorization
  - 009. Simple Security On Camera API
  - 010. Use A Queue On Camera Integration Service Container
  - 011. Integration Containers Database
  - 012. Manual Process On Cameras
  - 013. Mobile User Interface
  - 014. User Interface
- Solution
  - FirstIteration
    - C4Models
      - Legend
      - System Context Diagram
      - Container Diagram
      - Individual Container Diagram
        - Auth Service
        - Labelling Platform Integration
        - Dataset Manager
        - ML Training Manager
        - iNaturalist Integration
        - GBIF Integration
        - Camera Metadata Manager
        - Camera Manager
        - User Preferences
        - Camera Integration Service
        - Notification Service
  - Suggestion - Second Iteration

Structure

- Problem Background
  - Business Goals Drivers
  - Requirements
  - Considerations & Data Criticality
  - Actors & Actions
  - RAID Log
  - Analyses Of 3rd Party Tools
  - Glossary
- Solution Background
  - Vision
  - Architecture Principles
  - Proposed Scenarios
- ARDs
  - 000. Use ADR
  - 001. ADR Template
  - 002. Build The System Using Iterations
  - 003. Camera Update Strategy

- ARDs
  - 000. Use ADR
  - 001. ADR Template
  - 002. Build The System Using Iterations
  - 003. Camera Update Strategy
  - 004. Type Of User Notification
  - 005. Choose The First Labelling Platform Integrat
  - 006. Choose The First Model Training Integration
  - 007. Choosing The Microservice Architecture
  - 008. Merge Authentication And Authorization
  - 009. Simple Security On Camera API
  - 010. Use A Queue On Camera Integration Service Container
  - 011. Integration Containers Database
  - 012. Manual Process On Cameras
  - 013. Mobile User Interface
  - 014. User Interface
- Solution
  - FirstIteration

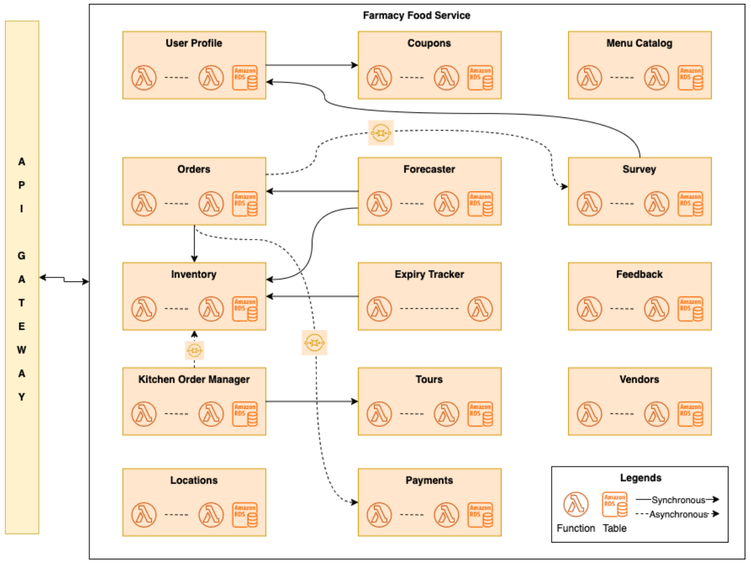
- Solution
  - FirstIteration
    - C4Models
      - Legend
      - System Context Diagram
      - Container Diagram
      - Individual Container Diagram
        - Auth Service
        - Labelling Platform Integration
        - Dataset Manager
        - ML Training Manager
        - iNaturalist Integration
        - GBIF Integration
        - Camera Metadata Manager
        - Camera Manager
        - User Preferences
        - Camera Integration Service
        - Notification Service
  - Suggestion - Second Iteration

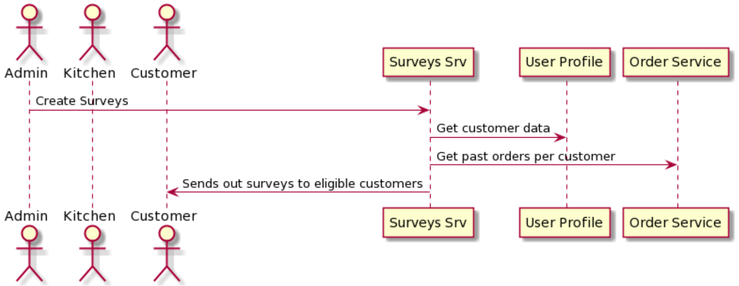
Structure

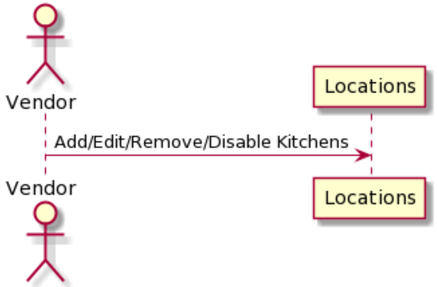
- Problem Background
  - Business Goals Drivers
  - Requirements
  - Considerations & Data Criticality
  - Actors & Actions
  - RAID Log
  - Analyses Of 3rd Party Tools
  - Glossary
- Solution Background
  - Vision
  - Architecture Principles
  - Proposed Scenarios
- ARDs
  - 000. Use ADR
  - 001. ADR Template
  - 002. Build The System Using Iterations
  - 003. Camera Update Strategy
  - 004. Type Of User Notification
  - 005. Choose The First Labelling Platform Integrat
  - 006. Choose The First Model Training Integration
  - 007. Choosing The Microservice Architecture
  - 008. Merge Authentication And Authorization
  - 009. Simple Security On Camera API
  - 010. Use A Queue On Camera Integration Service Container
  - 011. Integration Containers Database
  - 012. Manual Process On Cameras
  - 013. Mobile User Interface
  - 014. User Interface
- Solution
  - FirstIteration
    - C4Models
      - Legend
      - System Context Diagram
      - Container Diagram
      - Individual Container Diagram
        - Auth Service
        - Labelling Platform Integration
        - Dataset Manager
        - ML Training Manager
        - iNaturalist Integration
        - GBIF Integration
        - Camera Metadata Manager
        - Camera Manager
        - User Preferences
        - Camera Integration Service
        - Notification Service
  - Suggestion - Second Iteration

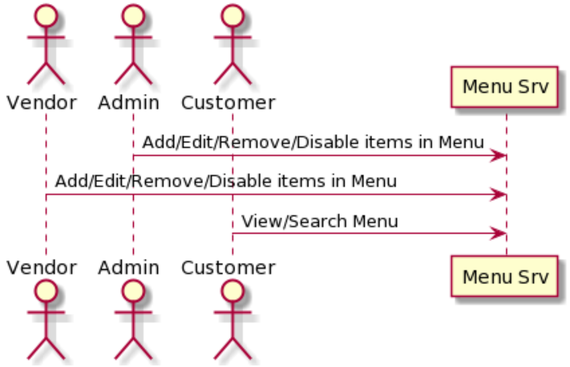
Completeness of solution

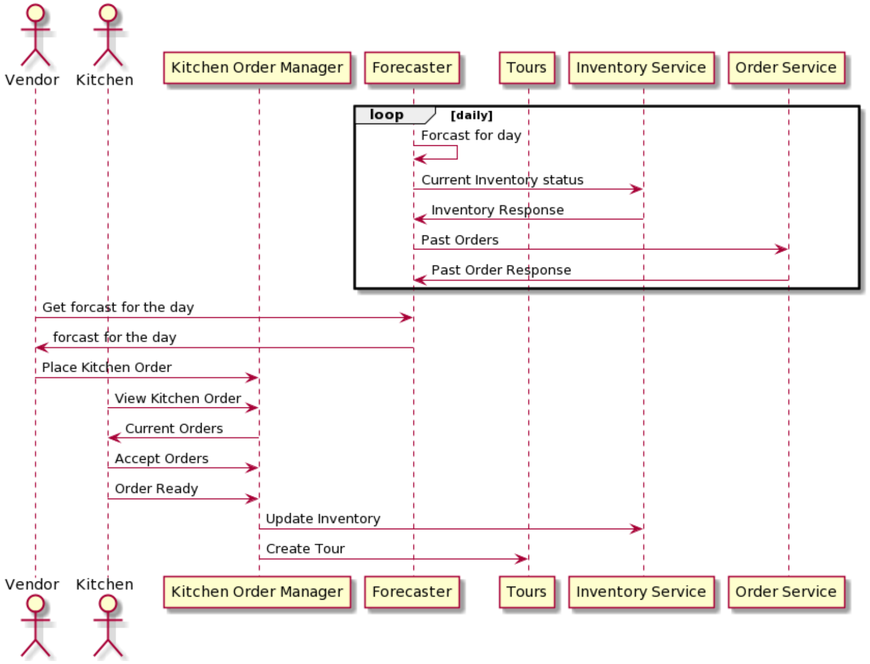
Completeness of Solution



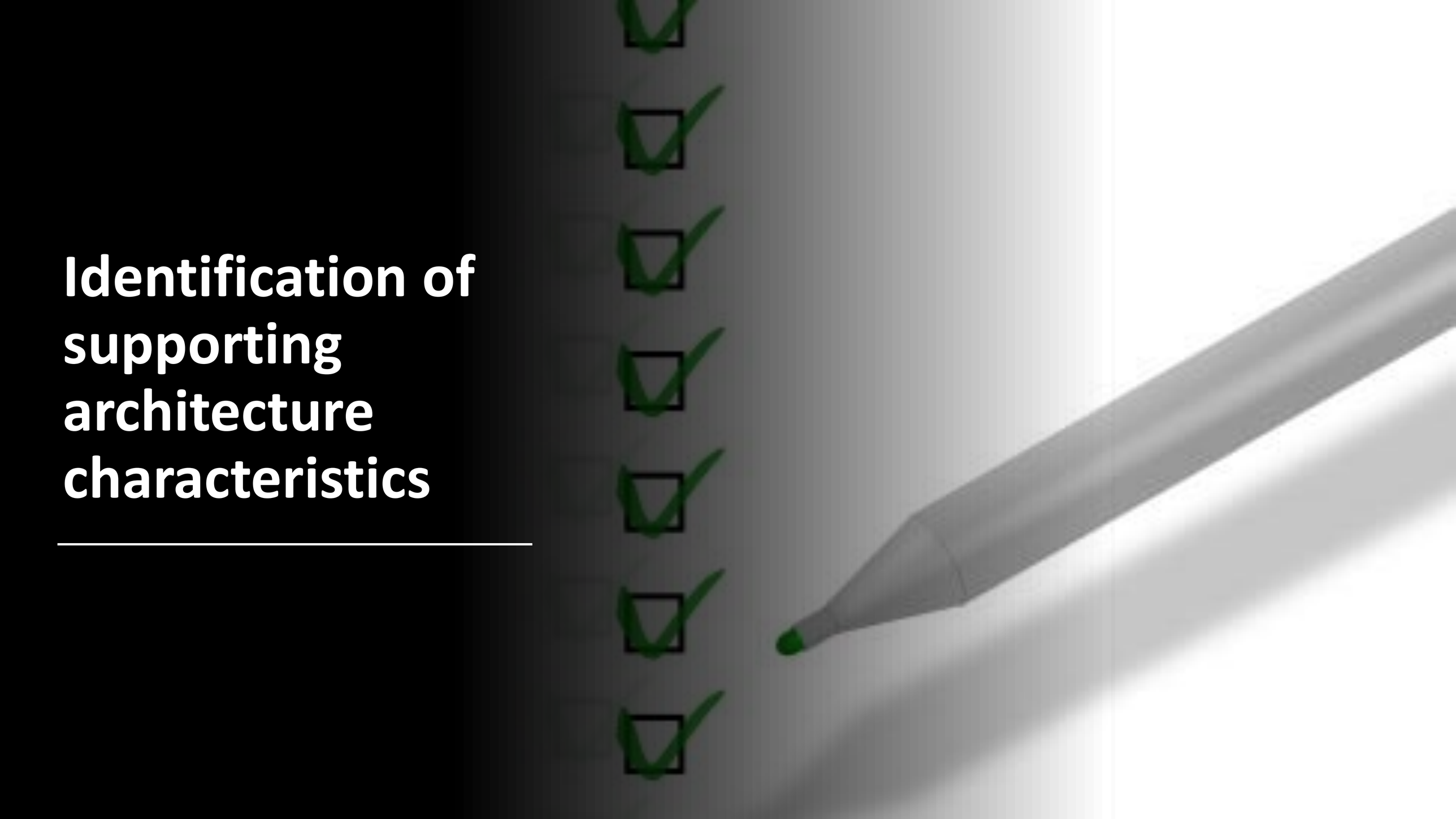






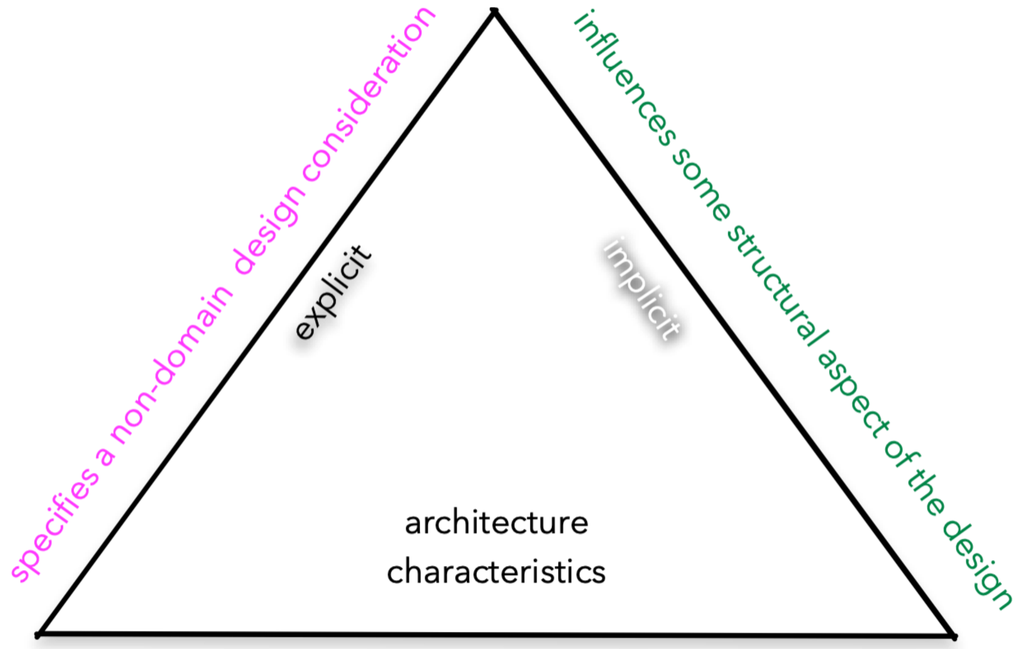


Identification of supporting architecture characteristics



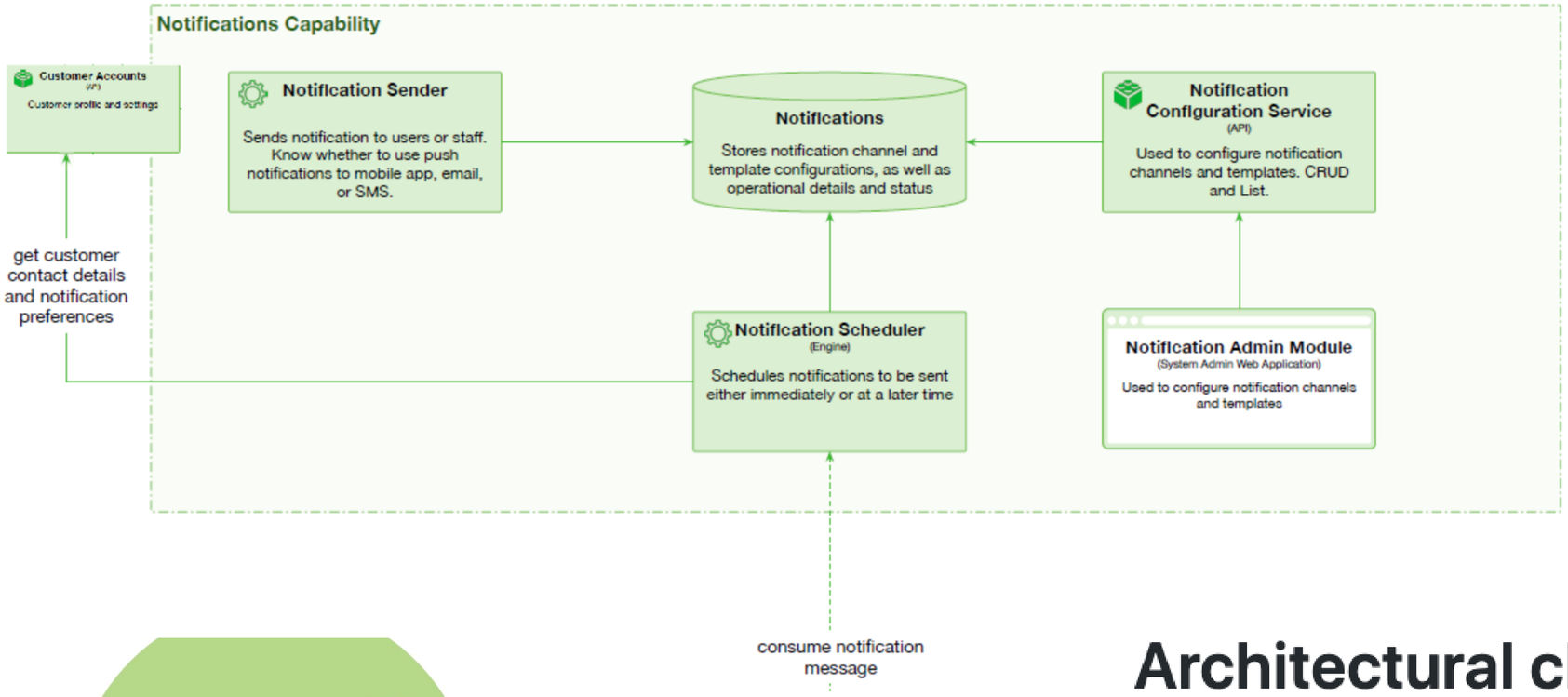
Architecture Characteristics

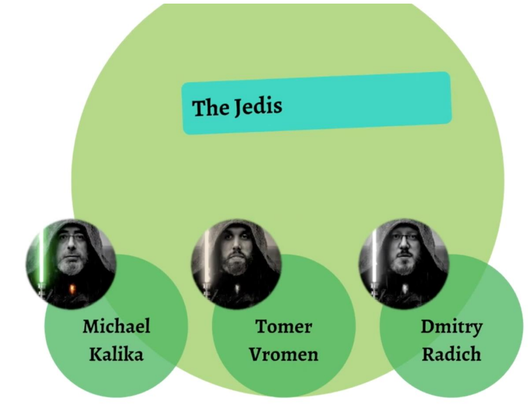
Architecture characteristics form the foundational aspects of the architecture and are required for proper trade-off analysis and decision making



- [x] Scope
- [x] Justification

Notifications Capability





Architectural characteristics

- Elasticity.
- Fault tolerance.
- Plugin support.

https://www.developertoarchitect.com/downloads/worksheets.html

https://www.developertoarchitect.com/lessons/lesson112.html

System/Project: MonitorMe

Domain: Hospital

Architect/Team: BluzBrothers

Date: 2024/02/17

Candidate Architecture Characteristics

performance	data integrity	deployability
responsiveness	data consistency	testability
availability	adaptability	abstraction
fault tolerance	extensibility	workflow
scalability	interoperability	configurability
elasticity	concurrency	recoverability

others: _____

_____

_____

a, b denotes characteristics that are related; some systems only need one of these, other systems may need both

Top 3	Driving Characteristics
☒	Performance
☐	Security
☐	Availability
☒	Evolvability
☒	Elasticity
☐	Fault-tolerance
☐	Configurability

Instructions

- Identify no more than 7 driving characteristics.
- Pick the top 3 characteristics (in any order).
- Implicit characteristics can become driving characteristics if they are *critical* concerns.
- Add additional characteristics identified that weren't deemed as important as the list of 7 to the *Others Considered* list.

Implicit Characteristics

feasibility (cost/time)

security

maintainability

observability

Others Considered

_____

_____

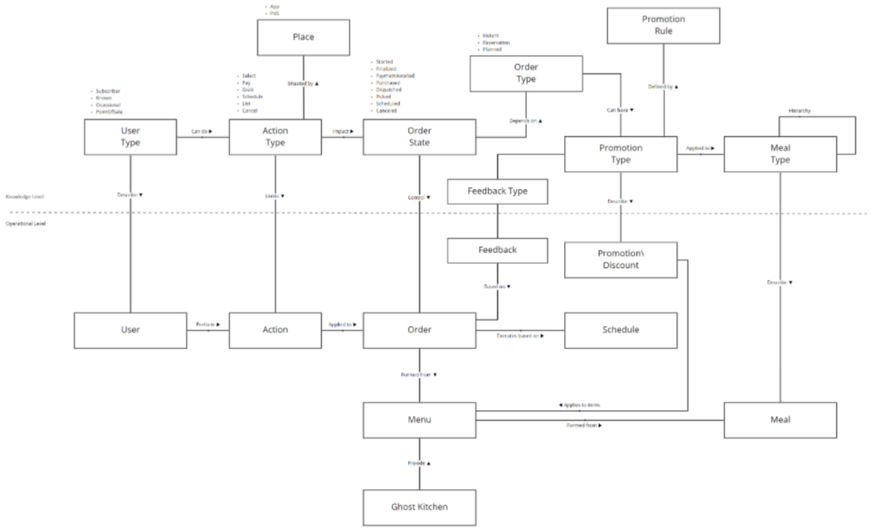
_____

_____

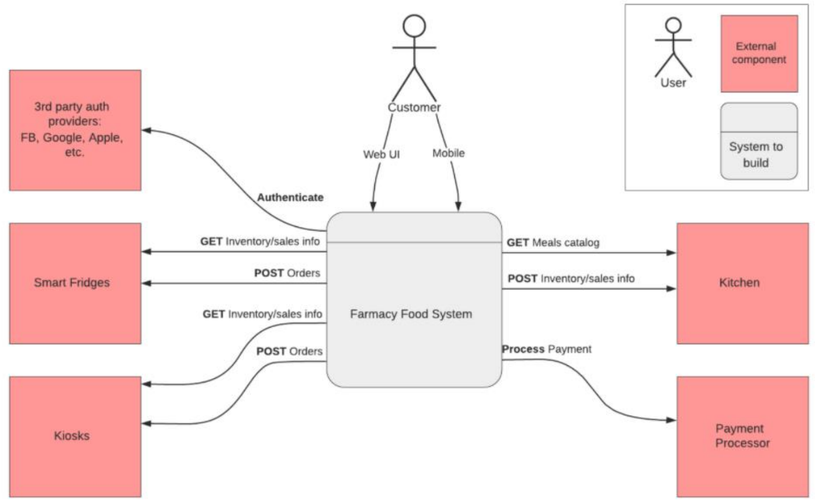
Diagrams – types, level of detail, and completeness

Diagrams

An effective architecture picture is worth more than a 1,000 words. Architecture represents topology, which benefits from visual representations.



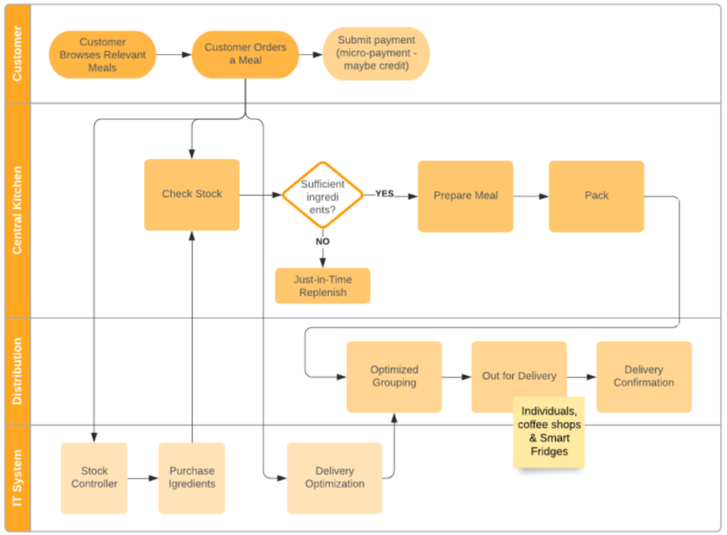
component diagrams



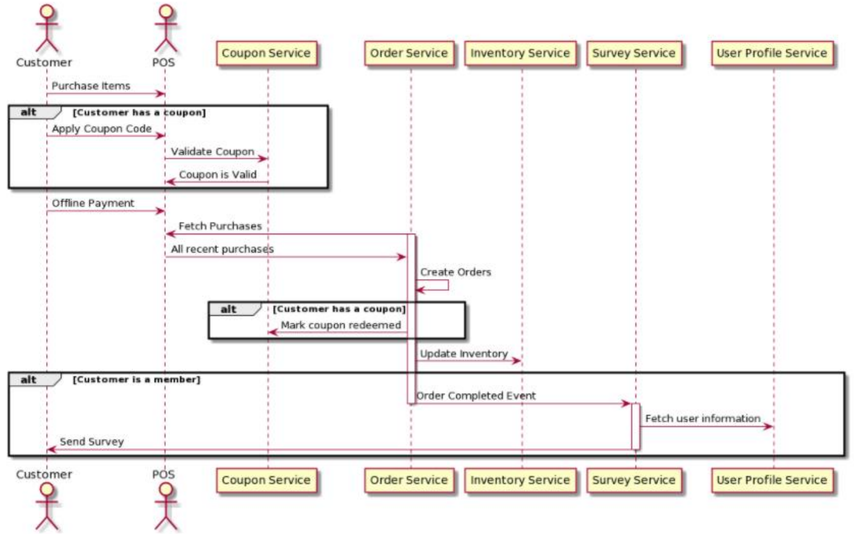
context diagrams

Diagrams

An effective architecture picture is worth more than a 1,000 words. Architecture represents topology, which benefits from visual representations.



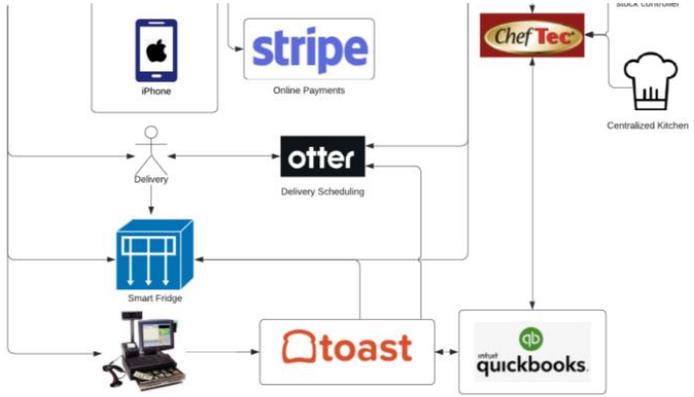
user journey diagrams



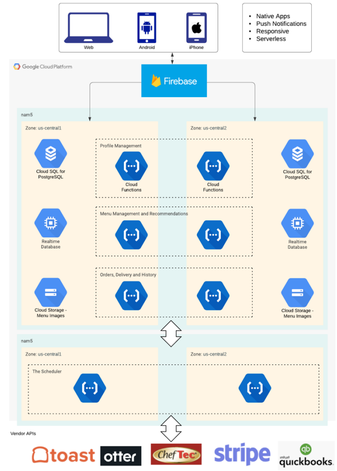
sequence diagrams

Diagrams

An effective architecture picture is worth more than a 1,000 words. Architecture represents topology, which benefits from visual representations.



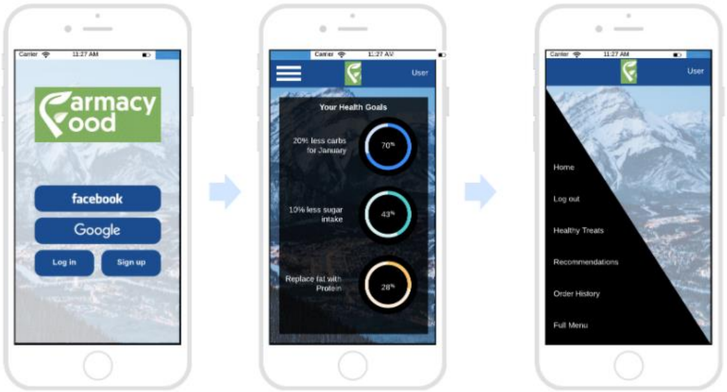
system-level diagrams



deployment diagrams

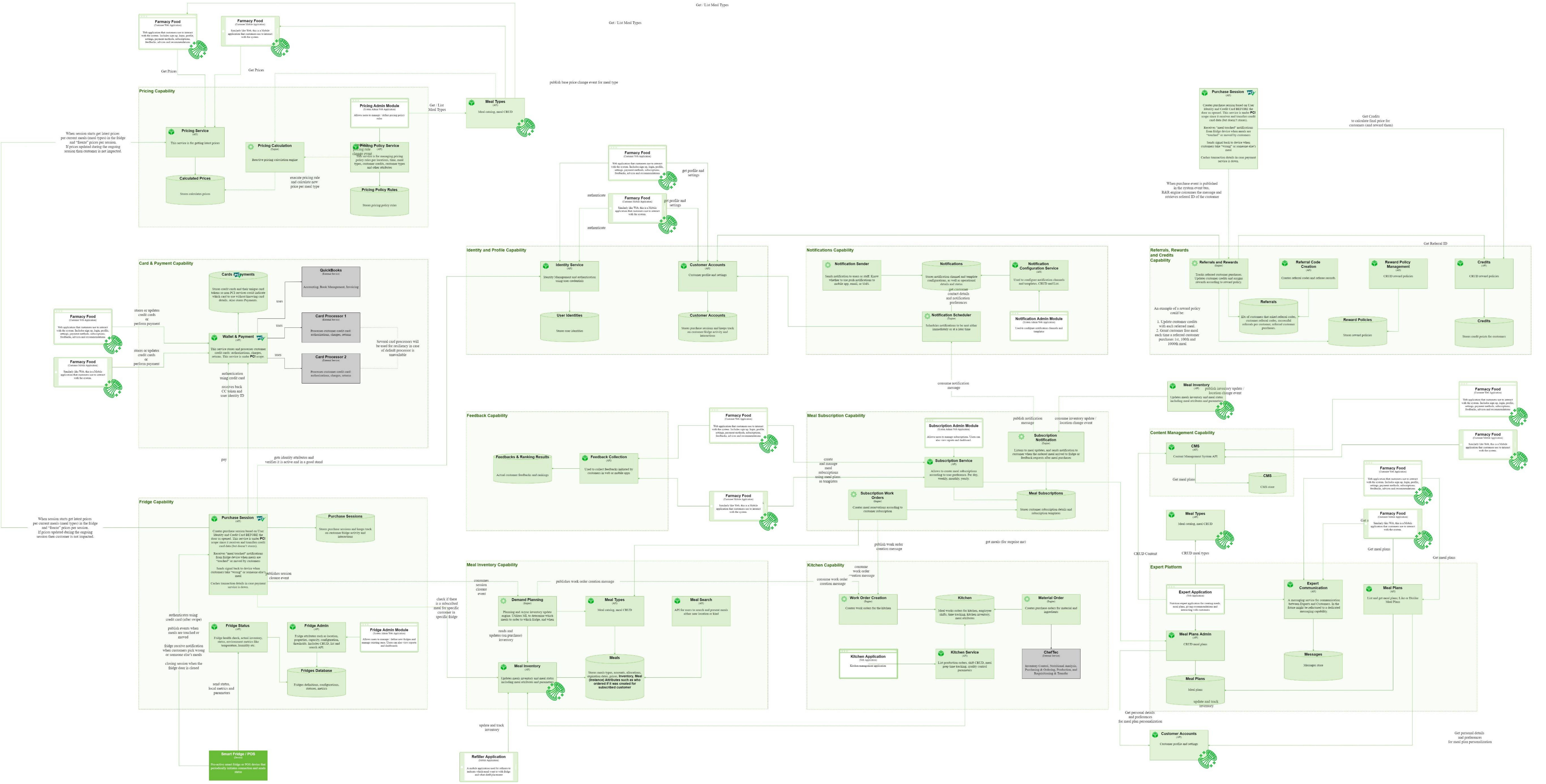
Diagrams

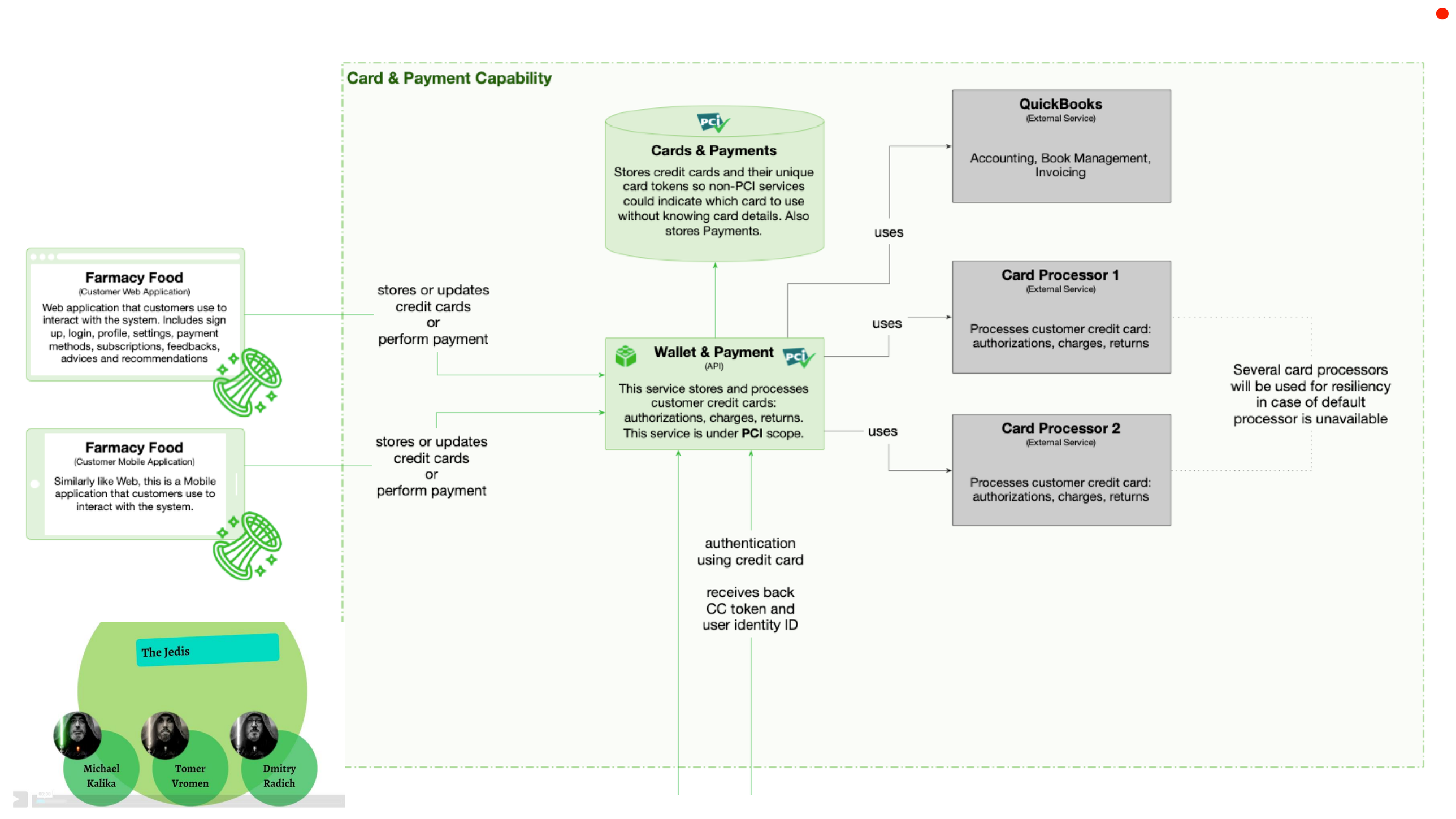
An effective architecture picture is worth more than a 1,000 words. Architecture represents topology, which benefits from visual representations.



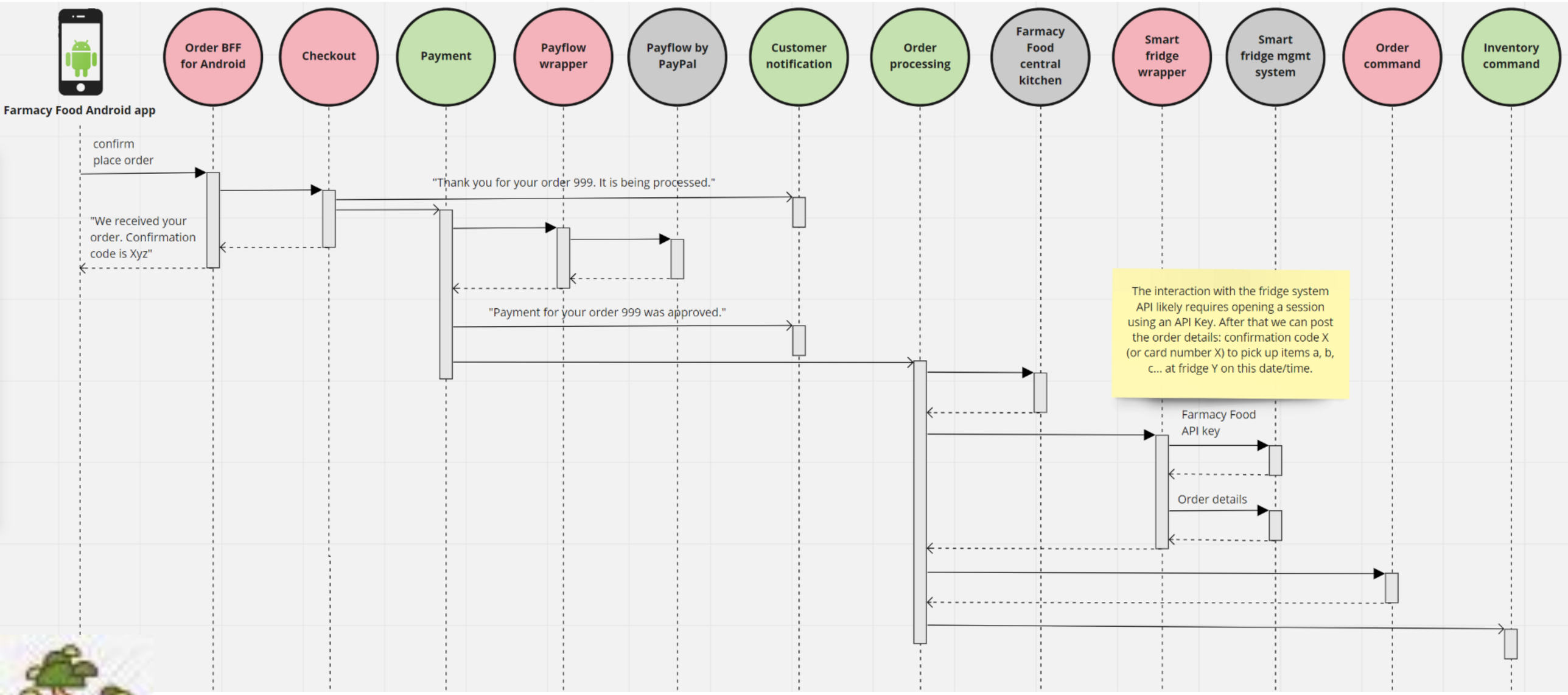
user interface mockups

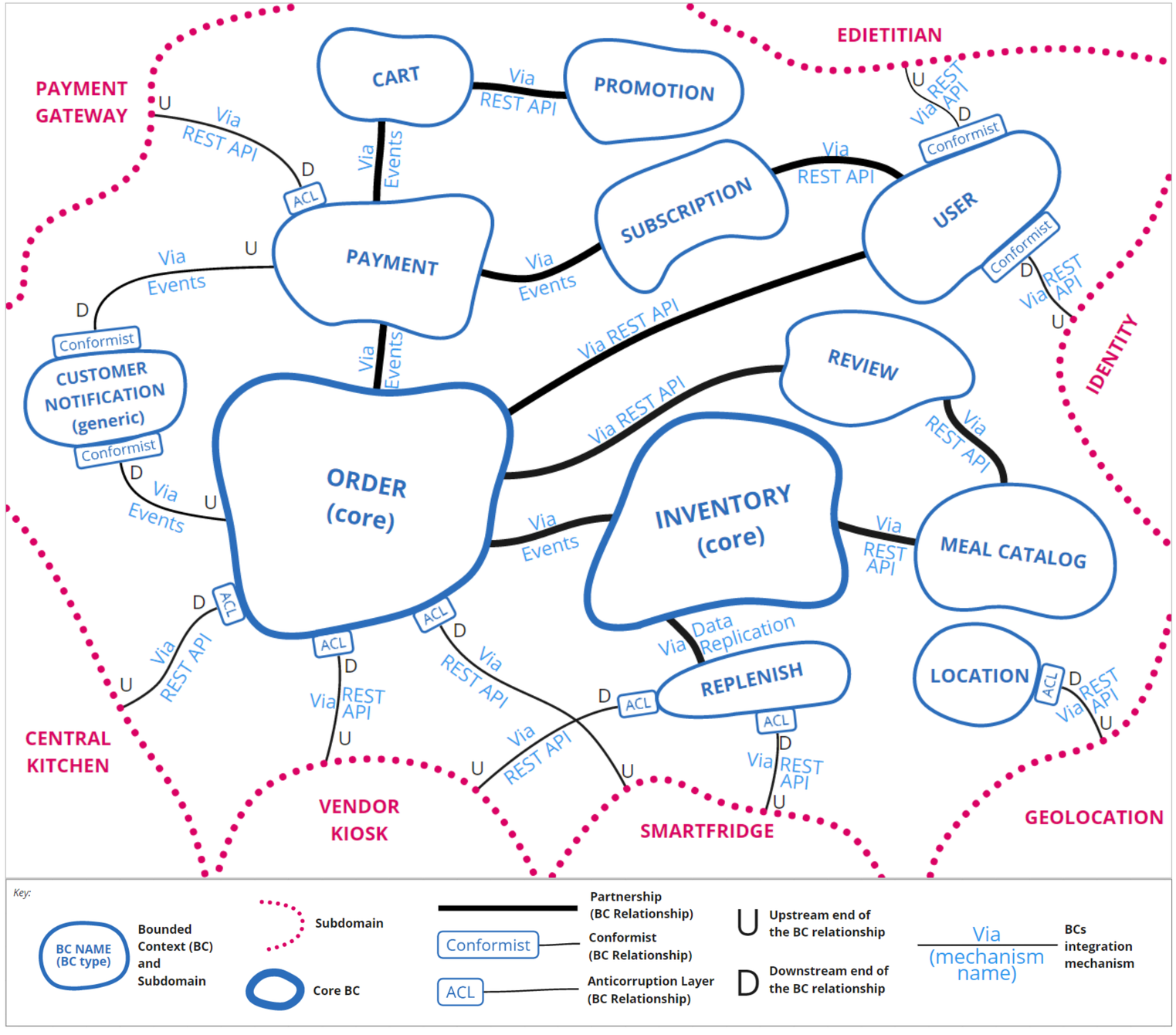
System Component Diagram




Card & Payment Capability
Cards & Payments
Stores credit cards and their unique card tokens so non-PCI services could indicate which card to use without knowing card details. Also stores Payments.
QuickBooks
(External Service)
Accounting, Book Management, Invoicing
uses
Card Processor 1
(External Service)
Processes customer credit card: authorizations, charges, returns
uses
Several card processors will be used for resiliency in case of default processor is unavailable
Farmacy Food
(Customer Web Application)
Web application that customers use to interact with the system. Includes sign up, login, profile, settings, payment methods, subscriptions, feedbacks, advices and recommendations
stores or updates credit cards or perform payment
Wallet & Payment
(API)
This service stores and processes customer credit cards: authorizations, charges, returns. This service is under PCI scope.
uses
Card Processor 2
(External Service)
Processes customer credit card: authorizations, charges, returns
Farmacy Food
(Customer Mobile Application)
Similarly like Web, this is a Mobile application that customers use to interact with the system.
stores or updates credit cards or perform payment
authentication using credit card
receives back CC token and user identity ID
The Jedis
Michael Kalika
Tomer Vromen
Dmitry Radich

Behavior


Farmacy Food Android app
Order BFF for Android
Checkout
Payment
Payflow wrapper
Payflow by PayPal
Customer notification
Order processing
Farmacy Food central kitchen
Smart fridge wrapper
Smart fridge mgmt system
Order command
Inventory command
confirm place order
"We received your order. Confirmation code is Xyz"
"Thank you for your order 999. It is being processed."
"Payment for your order 999 was approved."
The interaction with the fridge system API likely requires opening a session using an API Key. After that we can post the order details: confirmation code X (or card number X) to pick up items a, b, c... at fridge Y on this date/time.
Farmacy Food API key
Order details


PAYMENT GATEWAY
CART
PROMOTION
EDIETITIAN
SUBSCRIPTION
USER
IDENTITY
PAYMENT
CUSTOMER NOTIFICATION (generic)
ORDER (core)
REVIEW
INVENTORY (core)
MEAL CATALOG
REPLENISH
LOCATION
CENTRAL KITCHEN
VENDOR KIOSK
SMARTFRIDGE
GEOLOCATION
Via REST API
Via Events
Via Data Replication
Conformist
ACL
U
D
Key:
BC NAME (BC type)
Bounded Context (BC) and Subdomain
Subdomain
Core BC
Partnership (BC Relationship)
Conformist (BC Relationship)
Anticorruption Layer (BC Relationship)
Upstream end of the BC relationship
Downstream end of the BC relationship
Via (mechanism name)
BCs integration mechanism

https://www.developertoarchitect.com/downloads/worksheets.html

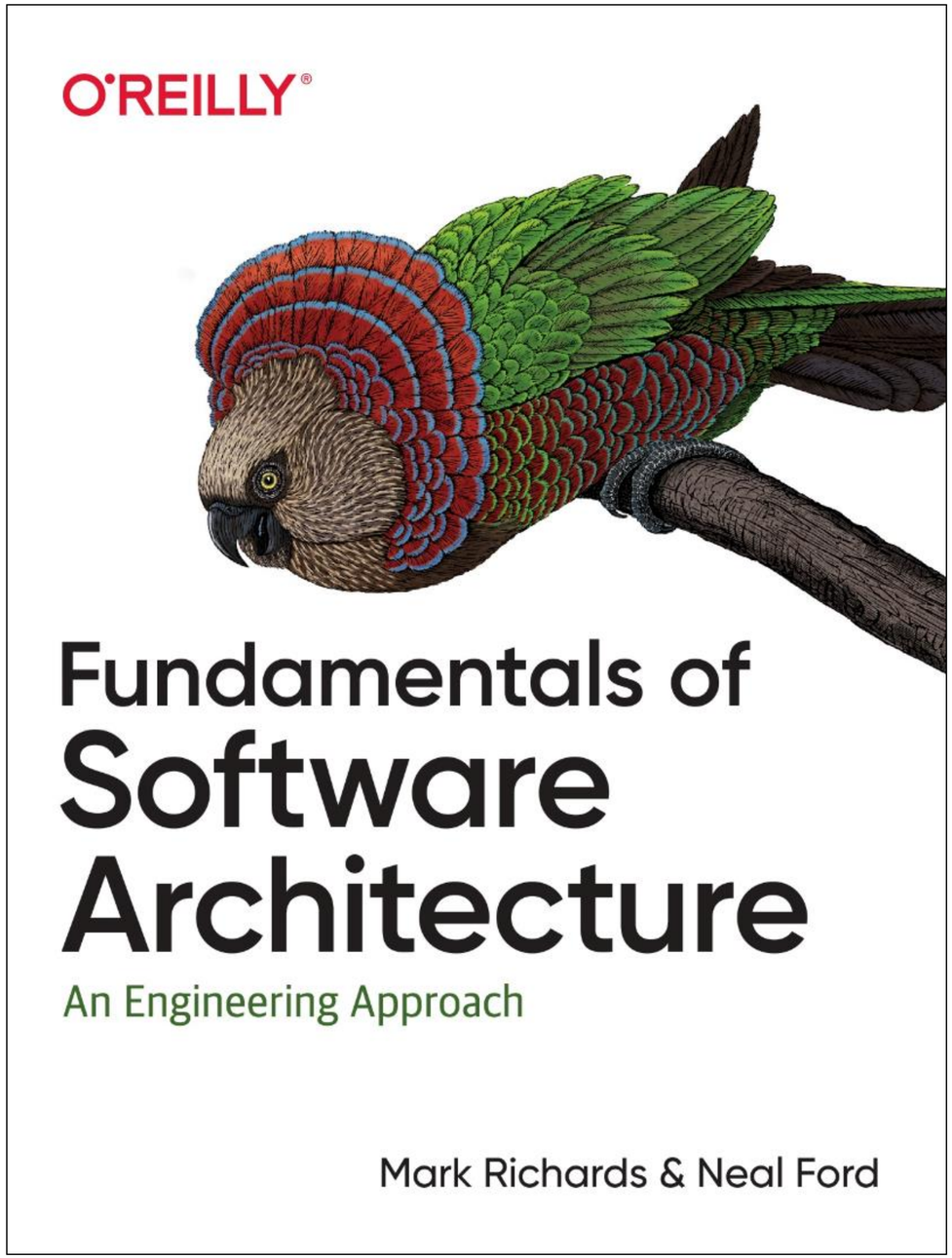
System/Project: MonitorMe

Architect/Team: BluzBrothers Date: 2024/02/17

Selected Architecture(s):

	☐ layered	☐ modular monolith	☐ microkernel	☐ microservices	☐ service-based	☐ service-oriented	☑ event-driven	☐ space-based
agility	★	★★	★★★	★★★★★	★★★★	★	★★★	★★
abstraction	★	★	★★★	★	★	★★★★★	★★★★	★
configurability	★	★	★★★★	★★★	★★	★	★★	★★
cost	★★★★★	★★★★★	★★★★★	★	★★★★	★	★★★	★★
deployability	★	★★	★★★	★★★★★	★★★★	★	★★★	★★★
domain part.	★	★★★★★	★★★★★	★★★★★	★★★★★	★	★	★★★★★
elasticity	★	★	★	★★★★★	★★	★★★	★★★★	★★★★★
evolvability	★	★	★★★	★★★★★	★★★	★	★★★★★	★★★
fault-tolerance	★	★	★	★★★★★	★★★★	★★★	★★★★★	★★★
integration	★	★	★★★	★★★	★★	★★★★★	★★★	★★
interoperability	★	★	★★★	★★★	★★	★★★★★	★★★	★★
performance	★★★	★★★	★★★	★★	★★★	★★	★★★★★	★★★★★
scalability	★	★	★	★★★★★	★★★	★★★	★★★★★	★★★★★
simplicity	★★★★★	★★★★★	★★★★	★	★★★	★	★	★
testability	★★	★★	★★★	★★★★★	★★★★	★	★★	★
workflow	★	★	★★	★	★	★★★★★	★★★★★	★

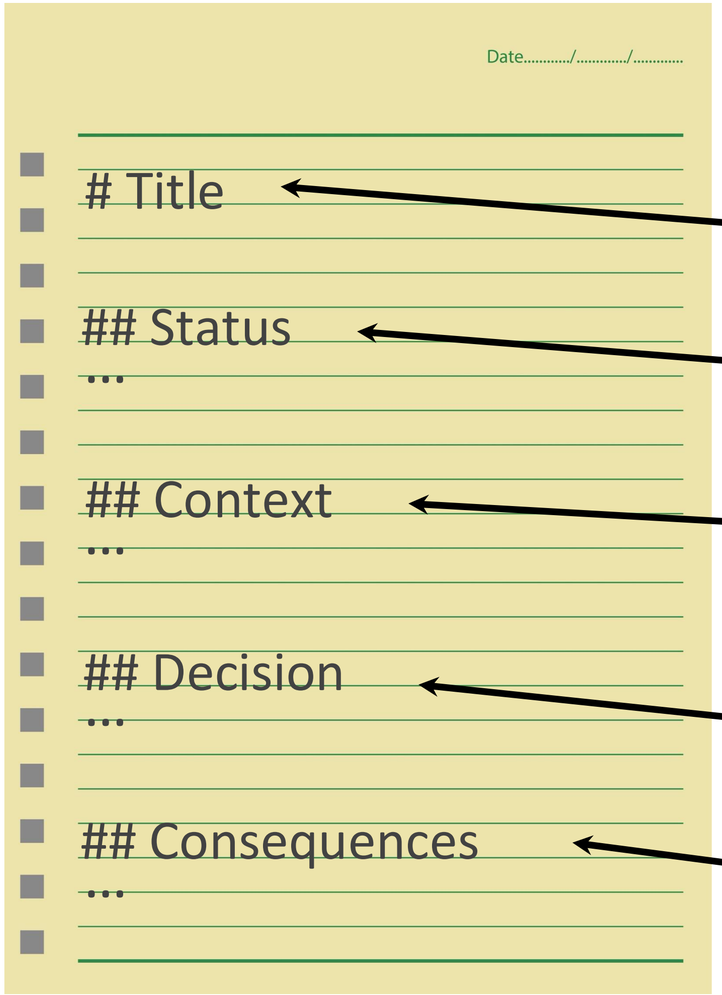
Architecture decision records – documentation and justification



Second Law of Software Architecture

"Why is more important then how"

architecture decision records



short text file; 1-2 pages long, one file per decision
markdown, textile, asciidoc, plaintext, etc.

short noun phrase

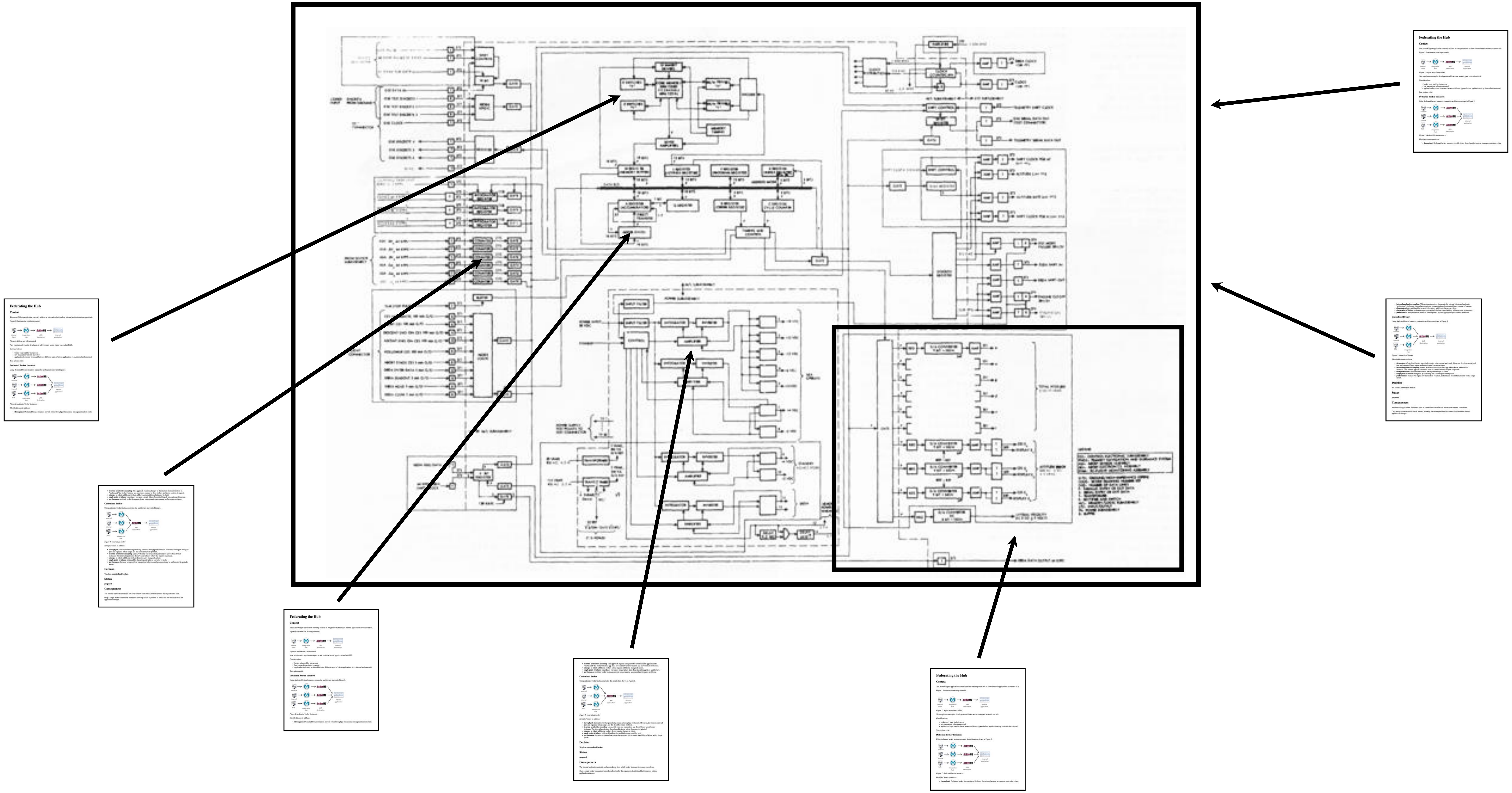
proposed, accepted, superseded

description of the problem and alternative solutions available (documentation)

decision and justification (the “why”)

trade-offs and impact of decision

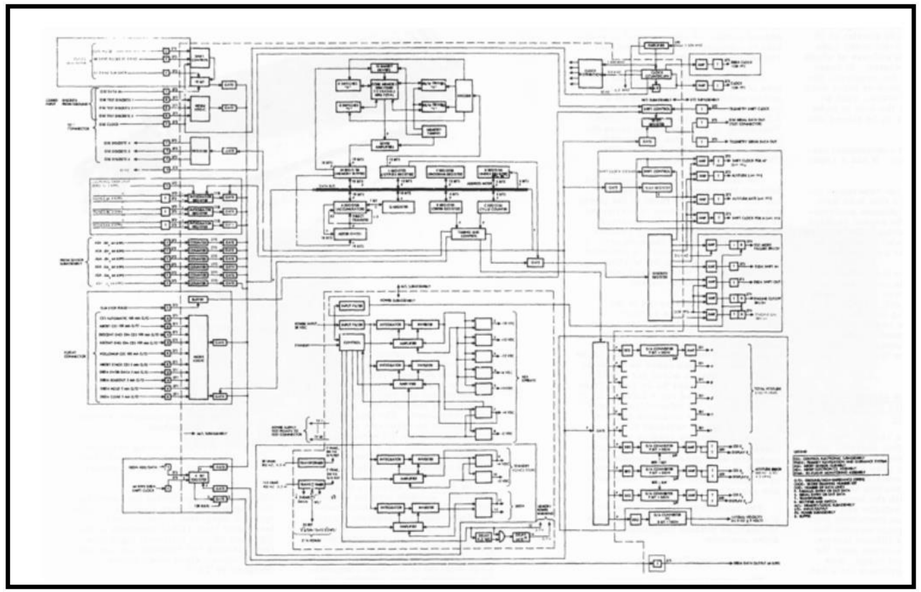
architecture decision records



architecture decision records

ADR 001: Use the microservice architecture style with containerization

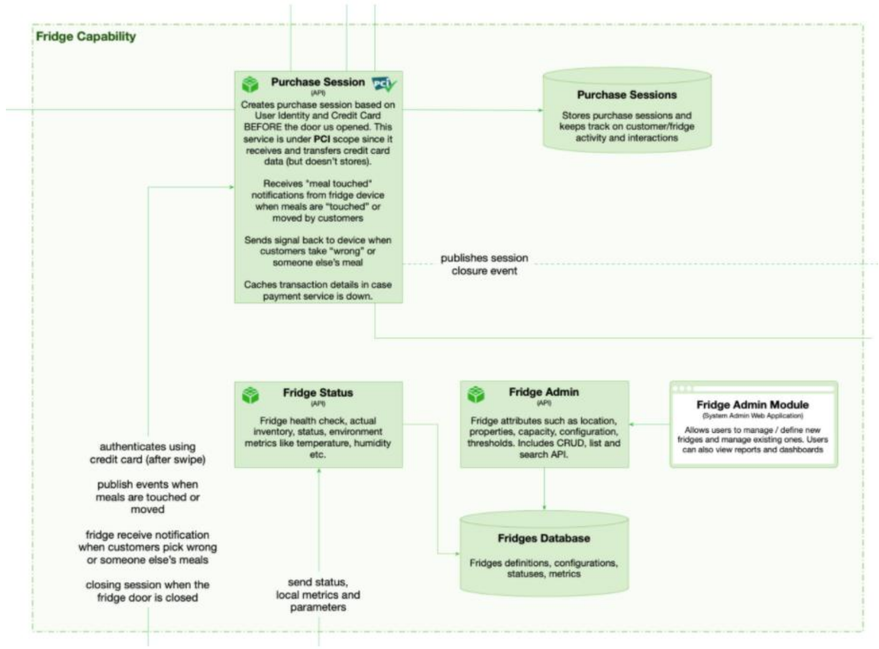
Farmacy Food is a start up company and does not have a sizeable team of experienced developers available. The overarching architecture style for the Farmacy Food system should be simple, easy to create, maintain and **evolve**. Finding developers that can create and evolve the system, as well as tools and frameworks that support the system should not require heaps of money. In other words, Farmacy Food is not in a position to be an *early adopter*, and should hence adopt an established architecture style that supports evolution.



Overall solution

Overall Solution

The architecture solution describes the overall structure of the system and how it will be constructed



- ☑ Are the architecture characteristics demonstrated in the solution?
- ☑ Is the solution appropriate and feasible given the project constraints?
- ☑ Are the architecture styles selected represented in the solution?

Integration

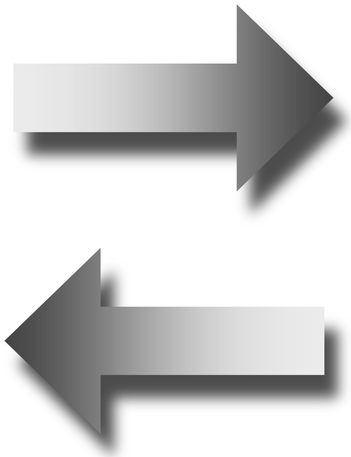
Feasibility

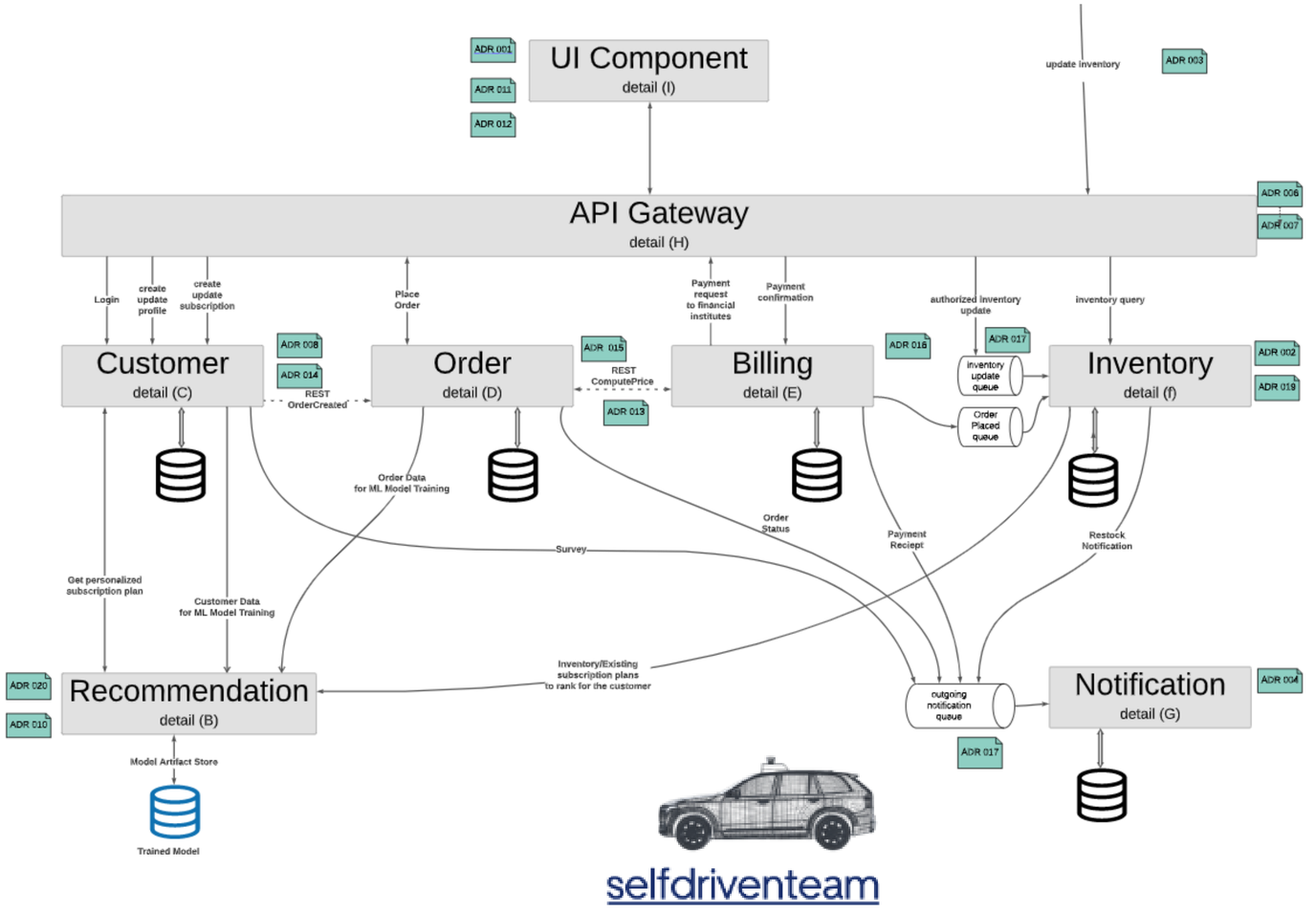
Agility

Availability

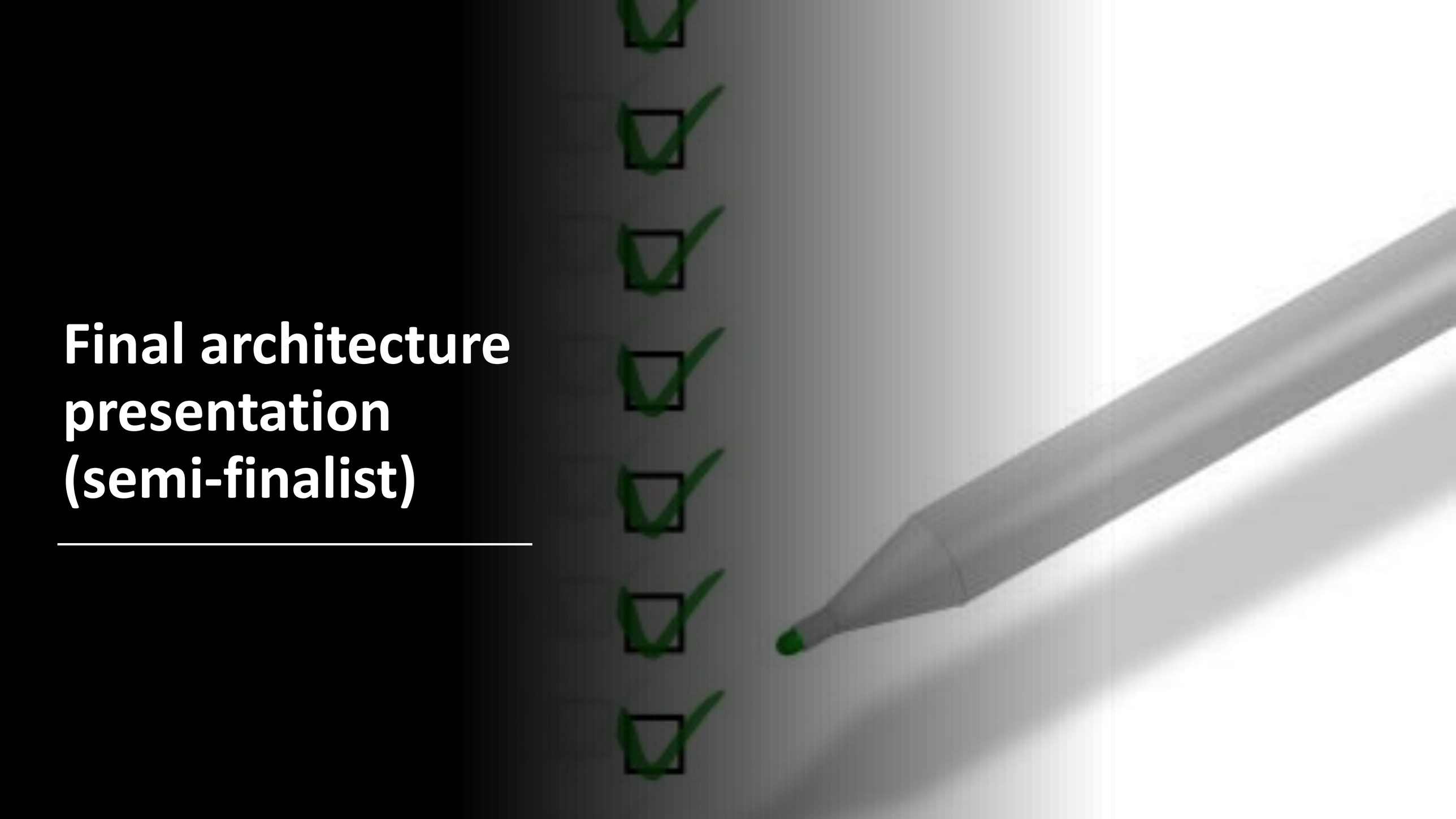
Security

Scalability





Final architecture presentation (semi-finalist)



The Business Problem

DIVERSITY CYBER COUNCIL
SECURING ACCESS TO OPPORTUNITY

https://diversitycybercouncil.com/

Diversity Cyber Council is a 501c3 Non-Profit that serves under-represented demographics in the tech industry by facilitating education, training, and staffing opportunities to establish a sustainable and diverse talent pipeline to the workforce

Diversity Cyber Council's goal is to establish a sustainable and diverse talent pipeline that extends career equity to underrepresented demographics by providing access to competent training programs that lead to direct employment opportunities.

DIVERSITY CYBER COUNCIL
SECURING ACCESS TO OPPORTUNITY

Industry Issues and Challenges

Lack of impactful metrics that identify and reduce potential biases in the job candidate hiring and interview process

Redundancy and ineffectiveness of the traditional applicant tracking software matching viable candidates with job descriptions

ClearView - Description

Clear View is a supplemental HR platform that anonymizes candidate information while highlighting objective skills and qualifying experience to reduce bias in the hiring process.

It leverages AI to construct stories about a job candidate based on S.M.A.R.T (Specific, Measurable, Achievable, Relevant, and Time-Bound) goals, qualifications, and experience, that are then quantifiably aligned with open roles.

All personal identifiable information and characteristics are eliminated until an objective determination is made on who the best candidate is to move forward with. The company pays to unlock the profile and data points are aggregated to reveal any disparities between those who are hired and those who were not selected.

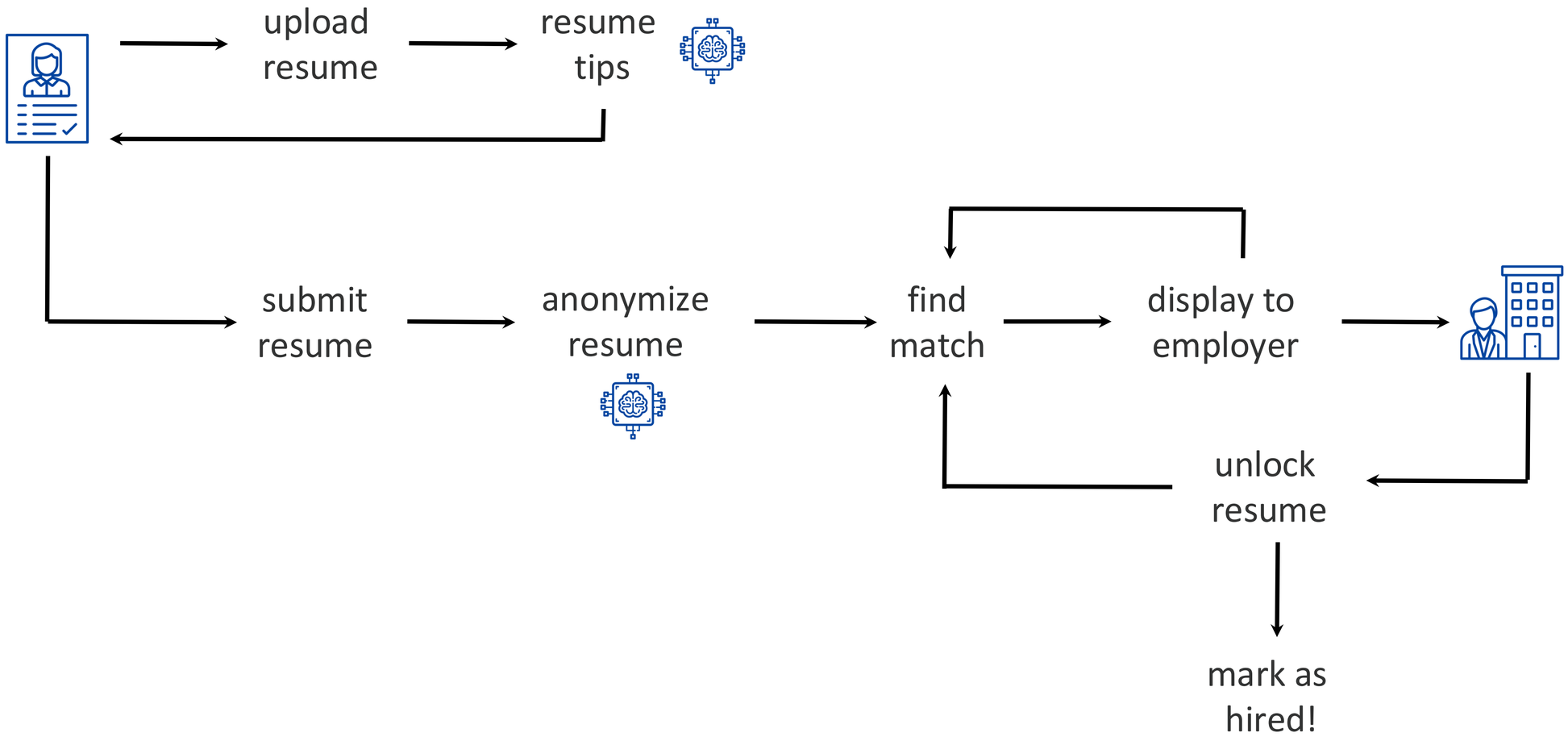
ClearView - User Roles

Employer
Companies invested in providing a more equitable experience to career seekers

Job Candidate
Professionals seeking a less tedious and more equitable hiring process that values their skills and abilities

Administrator
Management of the platform, registering users, providing data analytic reports on company performance and solution building services with executives

ClearView - Primary Workflow



ClearView - Context Diagram and Use Cases

Register
Upload resume
Update resume
Submit resume
View matches and unlocks
Remove resume
View employer information
Complete survey

Register
Enter company information (AI assisted)
Enter/update/ remove open job descriptions
View matches
Select candidate
Unlock resume
Mark candidate as hired
Mark candidate as passed (not hired)
Complete survey

Maintain users
Maintain internal information
Generate/view analytics
Generate/view reports

ClearView

Generate resume tips (AI-based)
Anonymize resume (AI-based)
Perform candidate matching
Invoicing for unlocking resume
Gather metrics and analytics

Unlocked full resume

HR System

HR System

HR System

. . .

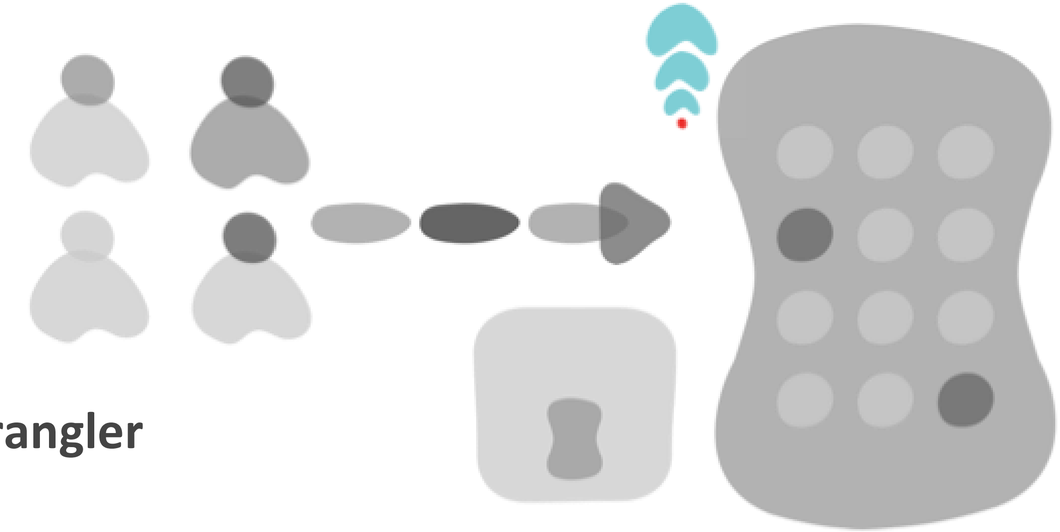
Contest Details

Dates

- All teams must submit a google form (https://forms.gle/fiRoB53SuUNMBNw98) by Friday, September 20 at 12PM Eastern to participate
- Solutions are due in your GitHub repo by Monday, September 30, 11:59PM Eastern
- Semifinalists will be announced at the second event on Thursday, October 10
- Questions? Email us at katas@oreilly.com

Architecture Katas

Autumn 2024: DCC



Neal Ford
Thoughtworks
Director / Software Architect / Meme Wrangler
https://www.nealford.com

Mark Richards
Independent Consultant
Hands-on Software Architect, Published Author
Founder, DeveloperToArchitect.com
@markrichardssa

Contest Kickoff