Application Modernization – End-to-End Example From Java EE in 2008 to Quarkus in 2021

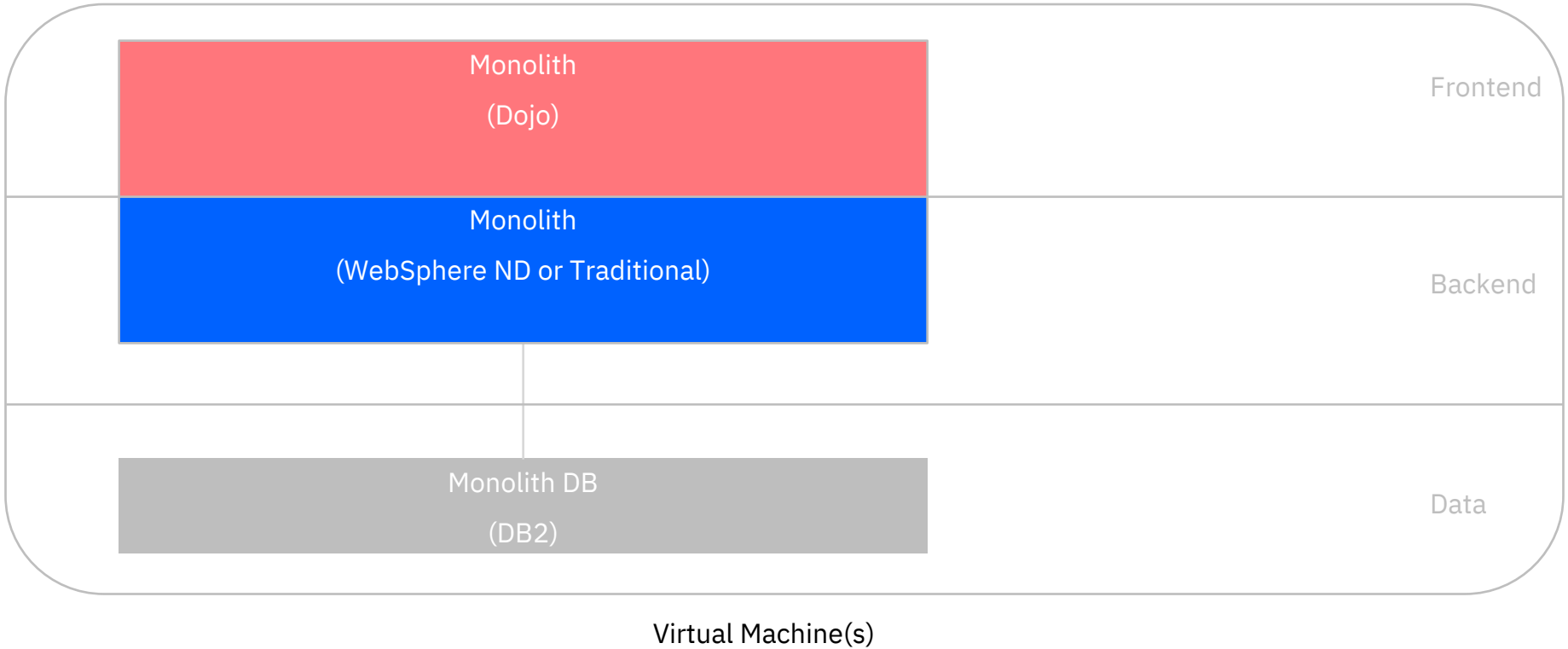
Niklas Heidloff
Developer Advocate, IBM
@nheidloff

Draft only
Work in progress ...

Step 1 / Starting Point: Monolith running in VMs

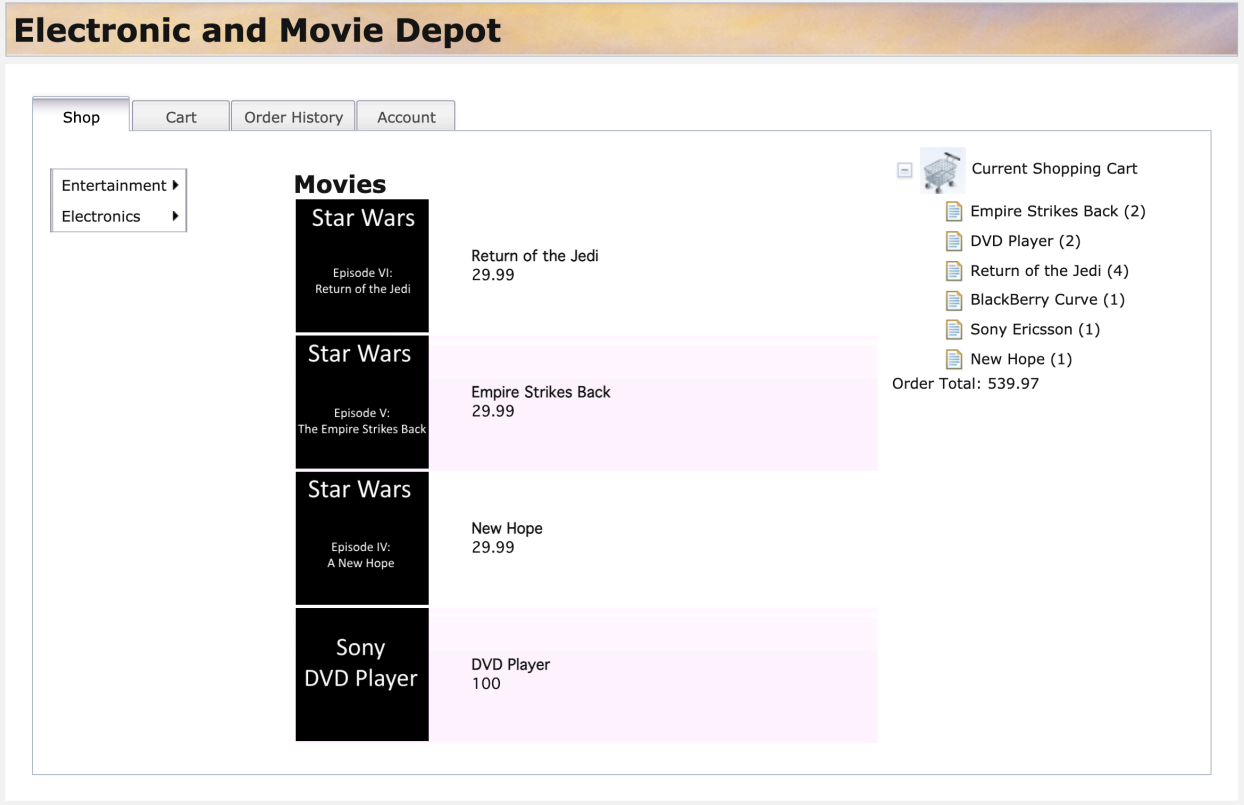
Description	How did we get here?	Used Technologies	Containers
Java EE 6 application Built around 2008 WebSphere 8.5.5 Typical three tier architecture No CI/CD	N/A – starting point	WebSphere ND or Traditional EJBs REST APIs (BFF, not RESTful) Db2 Dojo Toolkit	None Two Virtual Machines

Step 1 / Starting Point: Monolith running in VMs



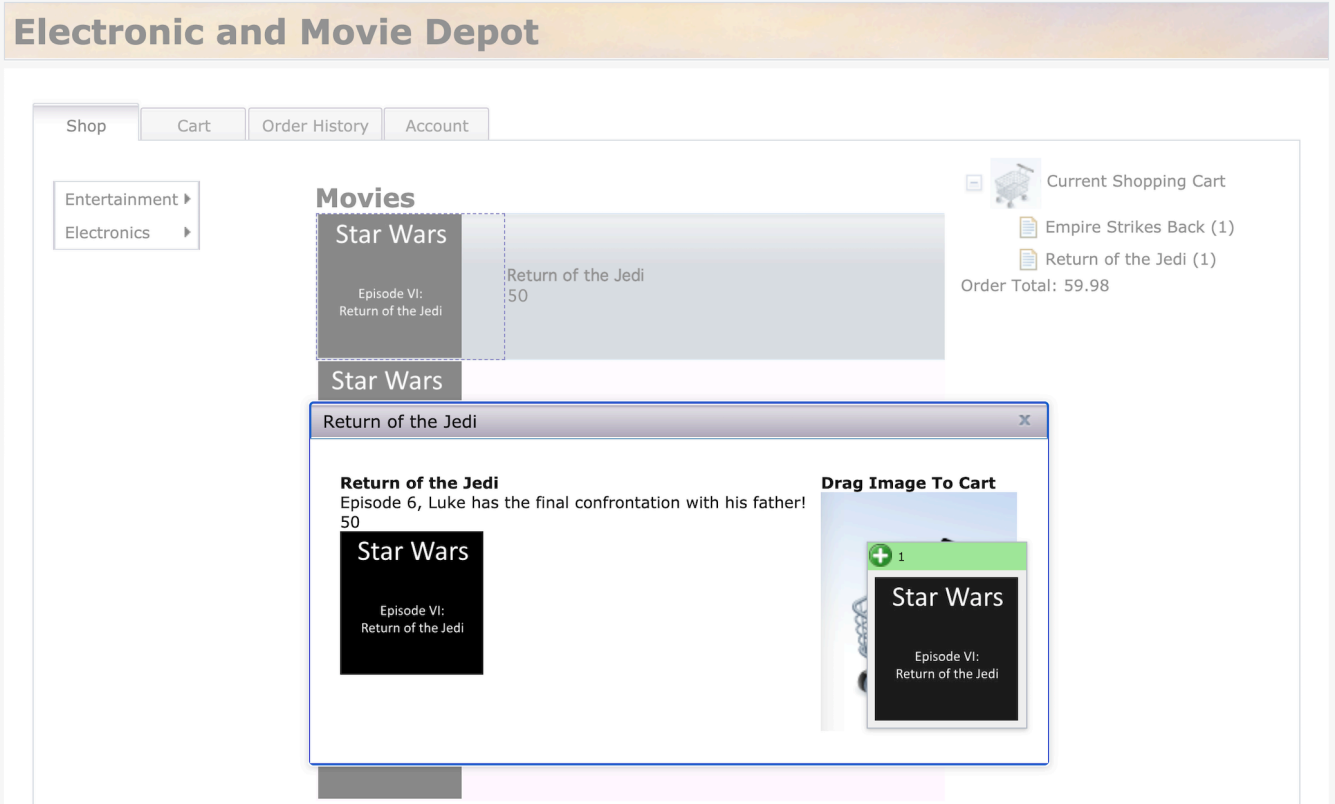
Step 1 / Starting Point: Monolith running in VMs

Functionality:
Browse Products



Step 1 / Starting Point: Monolith running in VMs

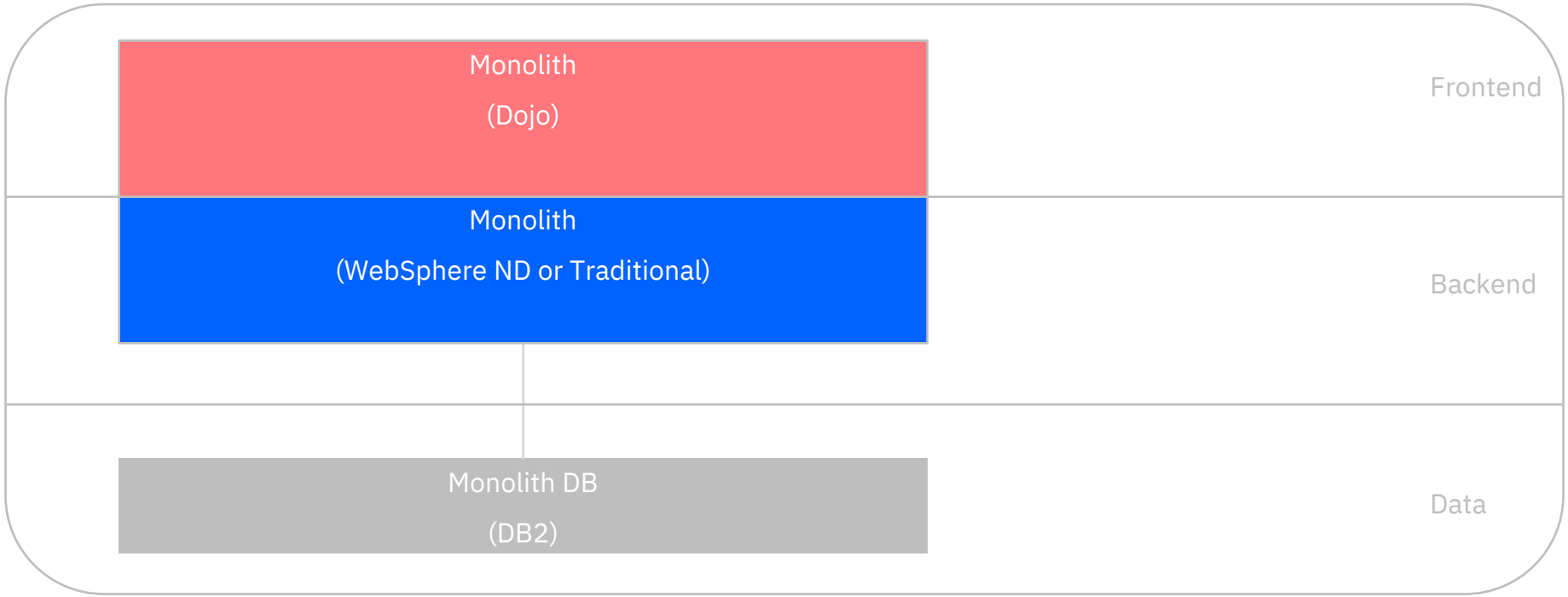
Functionality:
Add Products to Shopping Cart



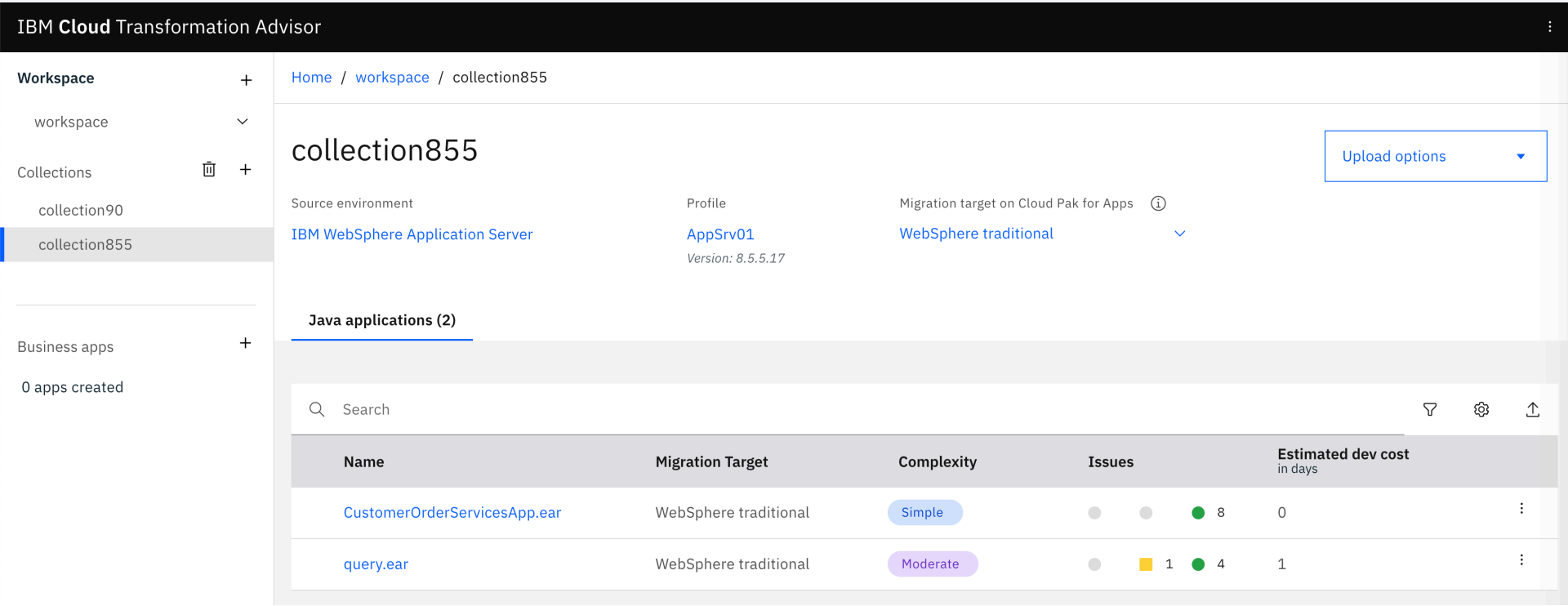
Step 2: Monolith running in Container on WebSphere Traditional

Description and Value	How did we get here?	Used Technologies	Containers
Monolith running in Container -> Can run on orchestration platforms like OpenShift -> Consistent management of all applications -> CI/CD possible	Transformation Advisor generated Dockerfile Manually created Db2 container	WebSphere Traditional 9	Monolith Db2

Step 2: Monolith running in Container on WebSphere Traditional



Step 2: Monolith running in Container on WebSphere Traditional



Step 2: Monolith running in Container on WebSphere Traditional

Migration Files

These files are generated by Transformation Advisor

server_config.py

install_app.py

Dockerfile

Pipeline resources

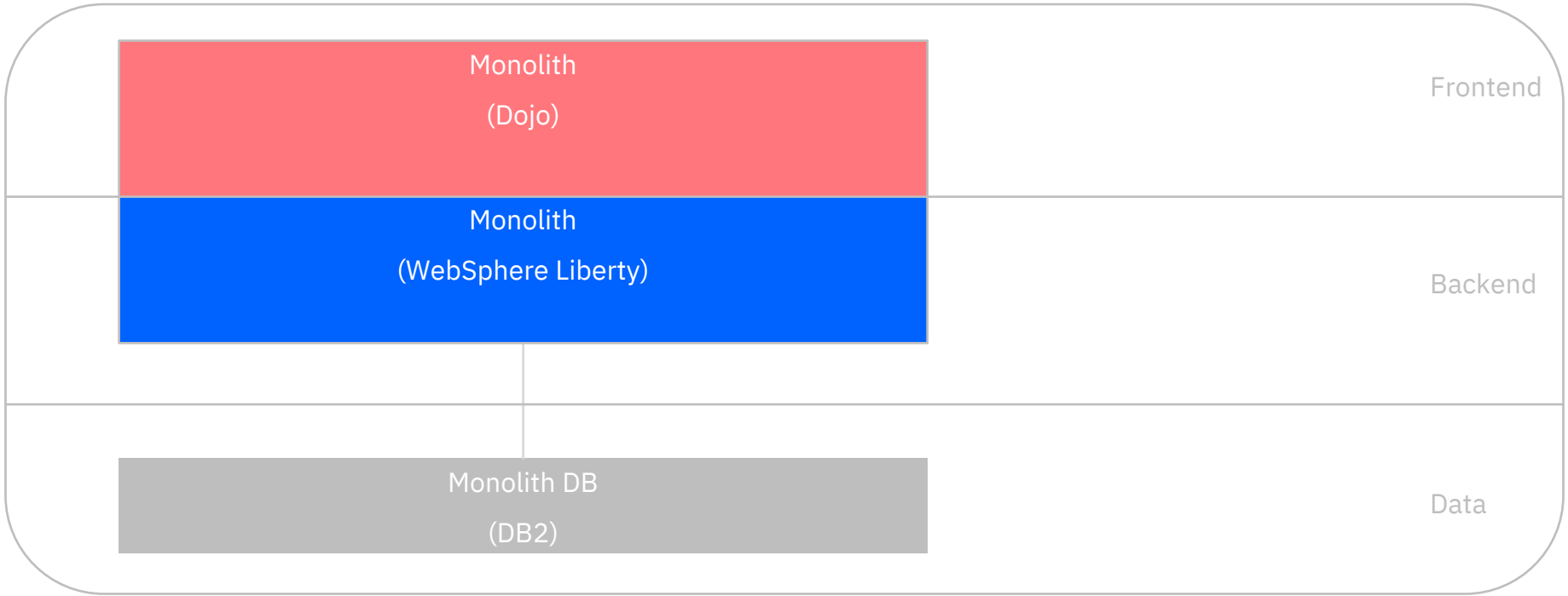
Transformation Advisor generates Dockerfile and scripts to run app in container

```
# Generated by IBM TransformationAdvisor
# Fri Jan 15 13:04:06 UTC 2021

# One manual change in line 12

FROM ibmcom/websphere-traditional:latest-ubi

# put app and scripts and properties in /work/config
# put external library (e.g db driver) in /work/config/lib

# Manual fix: Replace {APPLICATION_BINARY} with CustomerOrderServicesApp-0.1.0-SNAPSHOT.ear
COPY --chown=was:root CustomerOrderServicesApp-0.1.0-SNAPSHOT.ear /work/config/CustomerOrderServicesApp-0.1.0-SNAPSHOT.ear
COPY --chown=was:root ./src/config /work/config
COPY --chown=was:root ./lib /work/config/lib
RUN /work/configure.sh
```

Step 3: Monolith running in Container on WebSphere Liberty

Description and Value	How did we get here?	Used Technologies	Containers
Monolith running on Liberty -> More efficient runtime for container workloads	Transformation Advisor generated Dockerfile and server.xml Eclipse Migration Tools for code changes (?)	WebSphere Liberty	Monolith Db2

Step 3: Monolith running in Container on WebSphere Liberty



Step 3: Monolith running in Container on WebSphere Liberty

Connectivity Rules Summary

Jump To Rule

This table summarizes the flagged connectivity rules for each Java archive. Select the links in the column header to view all detailed results for that rule. Select the number links within this table to view the detailed results for that specific Java archive.

	Databases	Enterprise information systems (EIS)	Java EE security	Java Message Service (JMS)	JavaMail server	Message-Driven Beans (MDB)	Remote EJB lookups	Remote EJB providers	Remote web services	Third-party security	Vendor specific messaging
+ CustomerOrderServicesApp.ear	1		3								

Detailed Results by Rule

Expand all | Collapse all

Severe Rules

WebSphere traditional to Liberty

Behavior change on lookups for Enterprise JavaBeans (4)

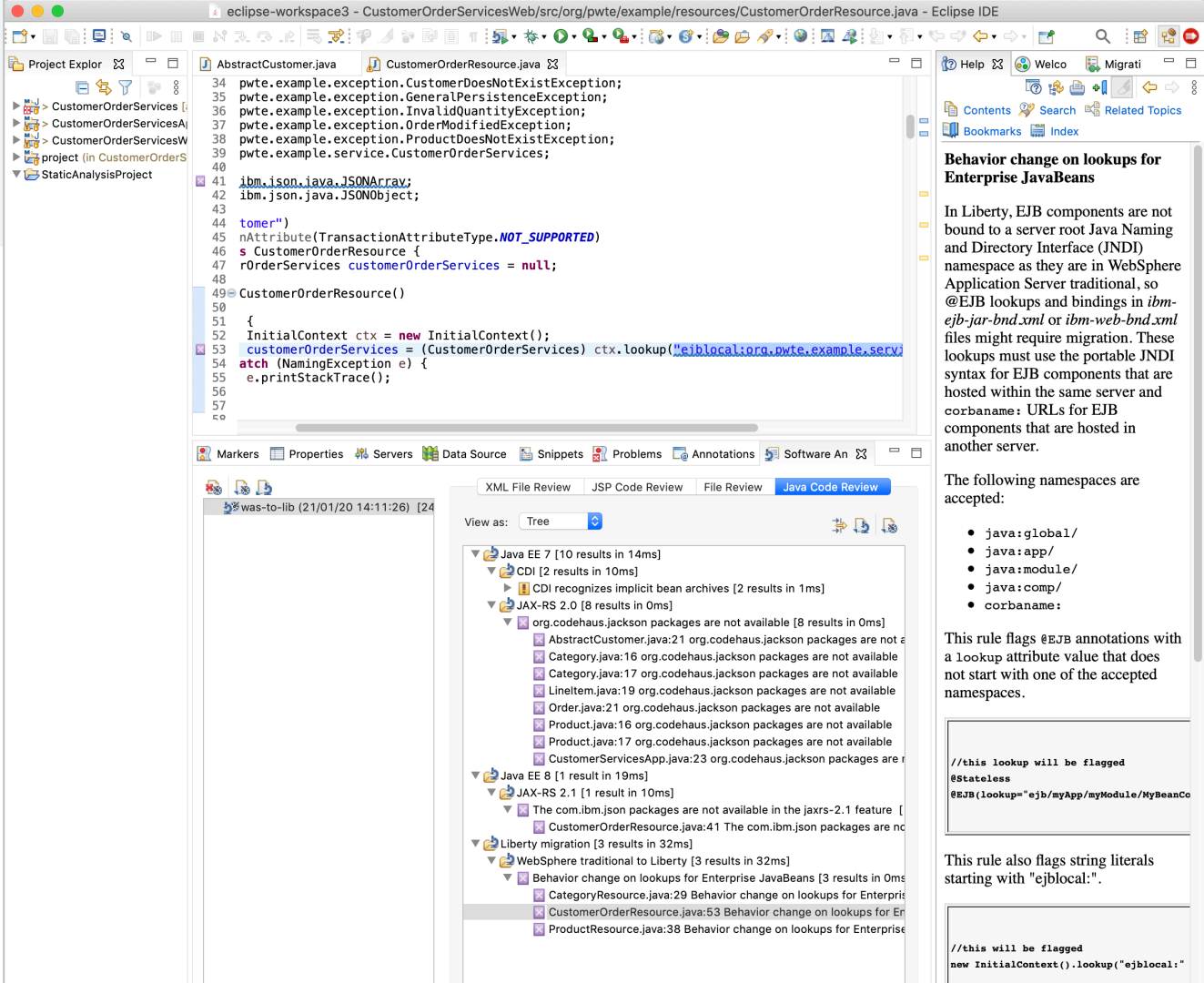
Show rule help Close results

Results

FILE NAME	REFERENCE DETAILS	MATCH CRITERIA	LINE NUMBER
CustomerOrderServicesApp.ear/CustomerOrderServicesTest-0.1.0-SNAPSHOT.war			
WEB-INF/ibm-web-bnd.xml		ejb-ref.binding-name=ejblocal:org.pwte.example.service.ProductSearchService	10
CustomerOrderServicesApp.ear/CustomerOrderServicesWeb-0.1.0-SNAPSHOT.war			
WEB-INF/classes/org/pwte/example/resources/CategoryResource.class	Constructor	ejblocal:org.pwte.example.service.ProductSearchService	28
WEB-INF/classes/org/pwte/example/resources/CustomerOrderResource.class	Constructor	ejblocal:org.pwte.example.service.CustomerOrderServices	53
WEB-INF/classes/org/pwte/example/resources/ProductResource.class	Constructor	ejblocal:org.pwte.example.service.ProductSearchService	38

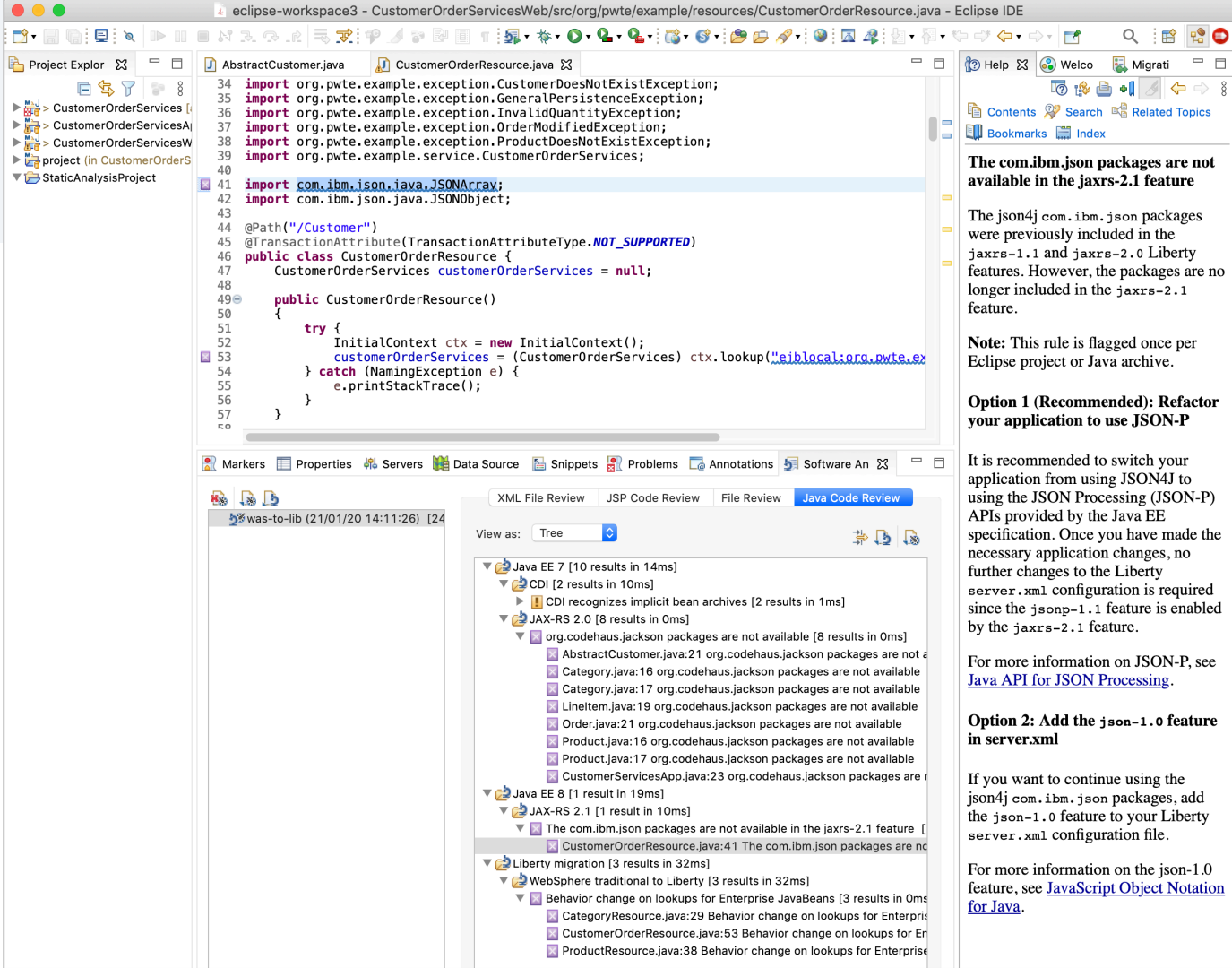
Step 3: WebSphere Liberty

Behavior change on lookups for EJBs



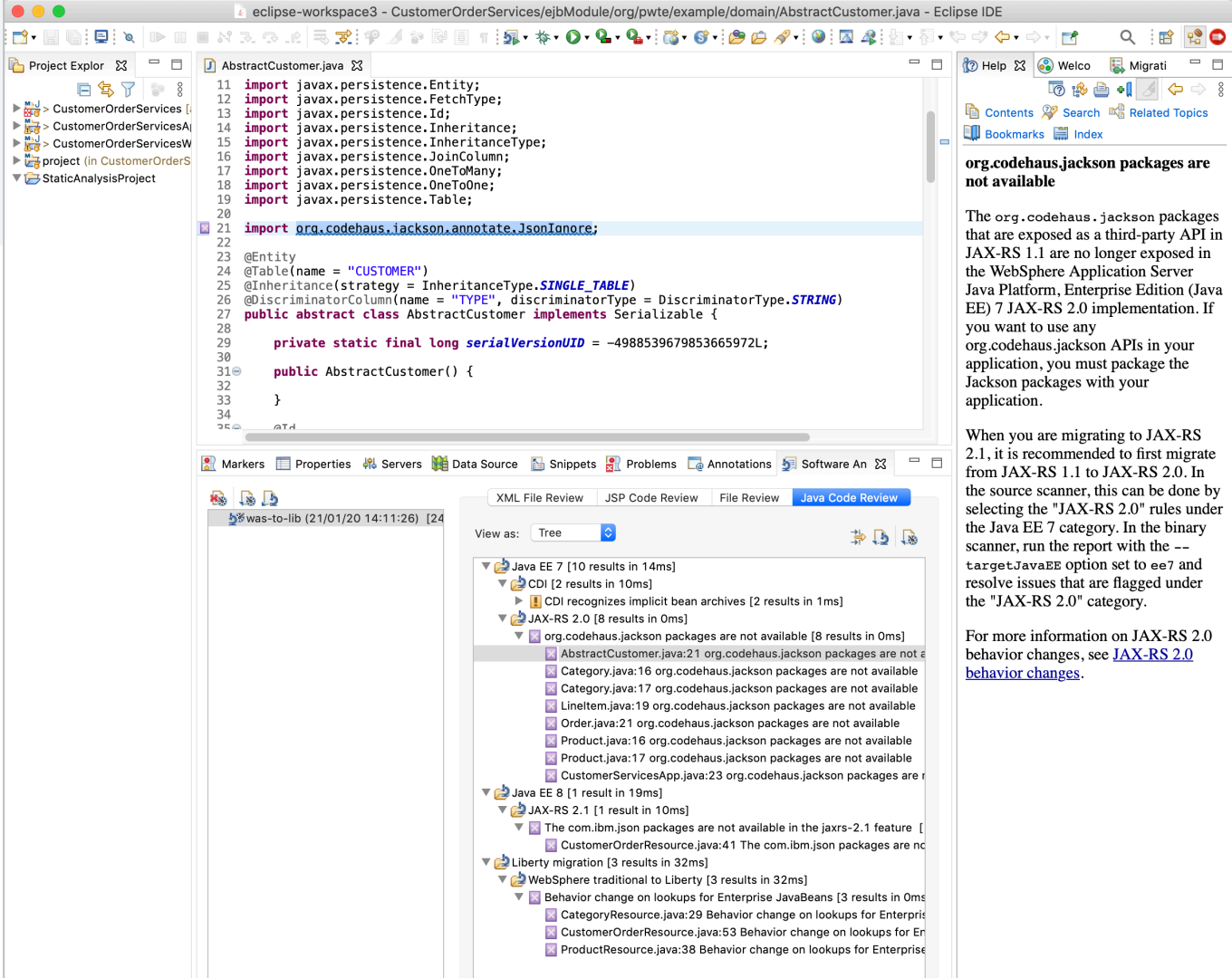
Step 3: WebSphere Liberty

com.ibm.json not available



Step 3: WebSphere Liberty

org.codehaus.jackson not available



Step 3: Monolith running in Container on WebSphere Liberty

Question:
Why doesn’t TA show all results that Migration Tools show?

Step 3: Monolith running in Container on WebSphere Liberty

Migration Files

These files are generated by Transformation Advisor

server.xml

pom.xml

Operator resources

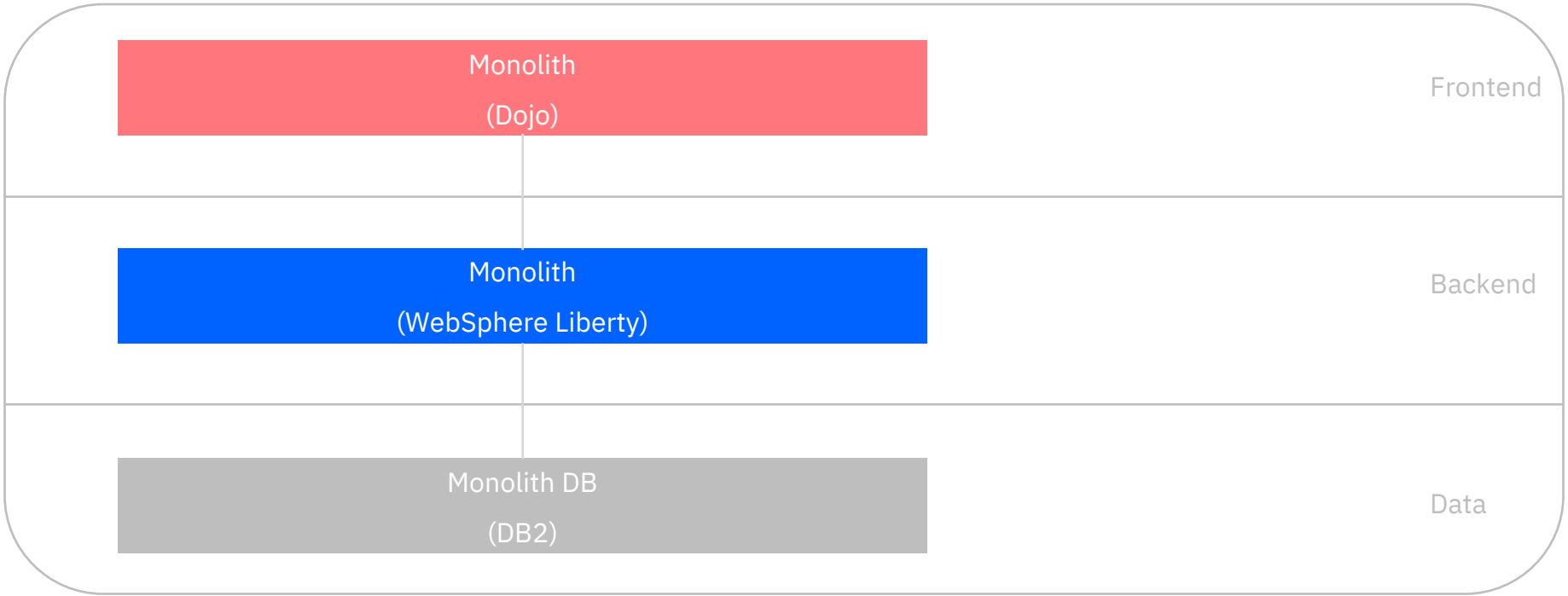
Dockerfile

Question:
Why don’t the generated files work?

Step 4: Separated Frontend

Description and Value	How did we get here?	Used Technologies	Containers
Separate container for Dojo frontend -> Easier extensions and potential replacement of frontend code	Manually	WebSphere Liberty Dojo	Backend monolith Frontend monolith Db2

Step 4: Separated Frontend



Step 4: Separated Frontend

For now the sample is run with Docker only

As ‘ingress’ an nginx proxy is used

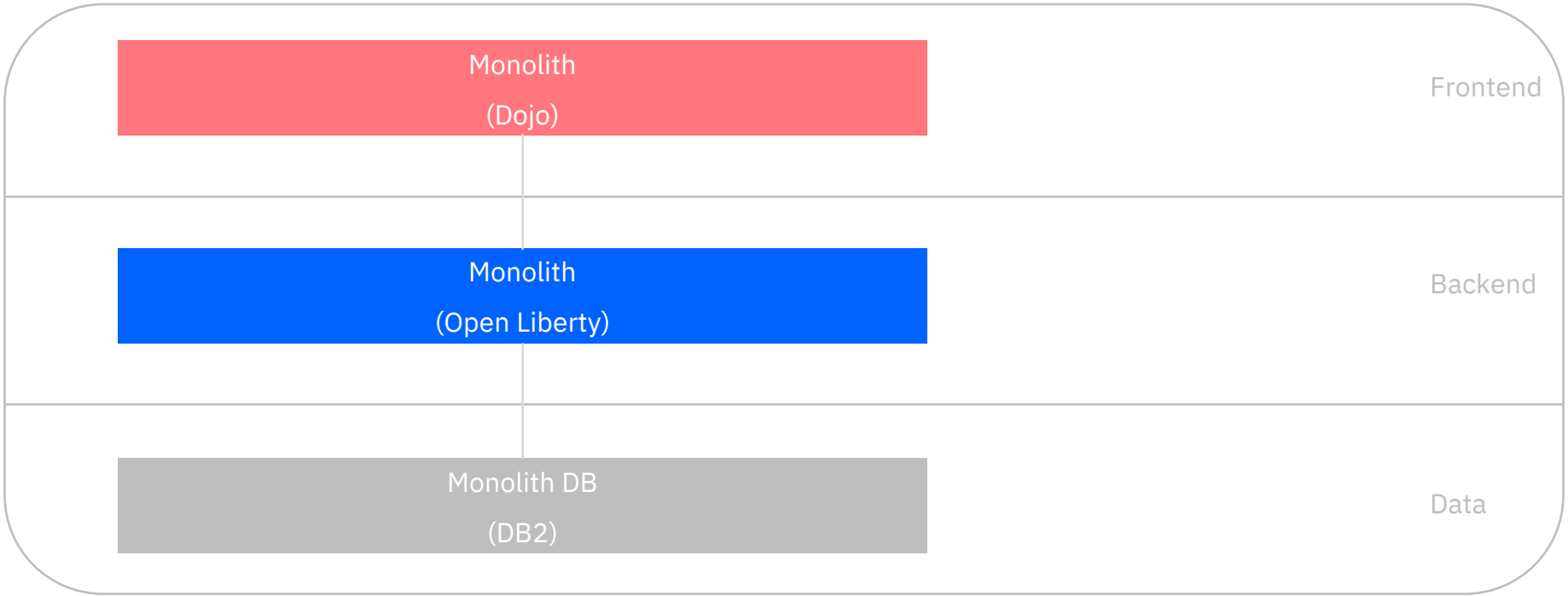
```yaml
version: '3.5'

networks:
  default:
    external:
      name: store-front-network

services:
    reverseproxy:
        image: proxy-nginx
        container_name: proxy-nginx
        ports:
            - 80:80
            - 443:443
        restart: always

    storefront-backend:
        image: storefront-backend
        container_name: storefront-backend
        restart: always
        environment:
          - CONTAINER=storefront-backend
        ports:
            - 9082:9080
            - 9445:9443

    storefront-frontend:
        image: storefront-frontend
        container_name: storefront-frontend
        restart: always
        environment:
          - CONTAINER=storefront-frontend
        ports:
            - 9081:9080
            - 9444:9443
```

```
worker_processes 1;

events { worker_connections 1024; }

http {

    sendfile on;

    upstream storefront-backend {
        server storefront-backend:9080;
    }

    upstream storefront-frontend {
        server storefront-frontend:9080;
    }

    server {
        listen 80;

        location / {
            proxy_pass         http://storefront-frontend;
            proxy_redirect     off;
            proxy_set_header   Host $host;
            proxy_set_header   X-Real-IP $remote_addr;
            proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header   X-Forwarded-Host $server_name;
        }

        location /CustomerOrderServicesWeb/jaxrs/ {
            proxy_pass         http://storefront-backend;
            proxy_redirect     off;
            proxy_set_header   Host $host;
            proxy_set_header   X-Real-IP $remote_addr;
            proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header   X-Forwarded-Host $server_name;
        }
    }
}
```

Step 5: Open Liberty Monolith

Description and Value	How did we get here?	Used Technologies	Containers
Usage of Open Liberty -> License cost savings -> Leverage latest MicroProfile and Open Liberty versions early and leverage open source communities	Manually (primarily changing image only)	Open Liberty MicroProfile	Backend monolith Frontend monolith Db2

Step 5: Open Liberty Monolith



Step 5: Open Liberty Monolith

Manually created Dockerfile

```
FROM adoptopenjdk/maven-openjdk11 as BUILD
COPY . /usr/src/app/src
WORKDIR /usr/src/app/src/CustomerOrderServicesProject
RUN mvn clean package

FROM open-liberty:kernel-slim-java11-openj9
USER root
COPY --chown=1001:0 ./liberty/server.xml /config
COPY --chown=1001:0 ./liberty/server.env /config
COPY --chown=1001:0 ./liberty/jvm.options /config

ARG SSL=false
ARG HTTP_ENDPOINT=false

RUN features.sh

COPY --from=build --chown=1001:0 /usr/src/app/src/CustomerOrderServicesApp/target/CustomerOrderServicesApp-0.1.0-SNAPSHOT.ear /config/apps/CustomerOrderServicesApp-0.1.0-SNAPSHOT.ear
COPY --from=build --chown=1001:0 /usr/src/app/src/resources/ /opt/ol/wlp/usr/shared/resources/
RUN chown -R 1001.0 /config /opt/ol/wlp/usr/servers/defaultServer /opt/ol/wlp/usr/shared/resources &&
    chmod -R g+rw /config /opt/ol/wlp/usr/servers/defaultServer  /opt/ol/wlp/usr/shared/resources

USER 1001
RUN configure.sh
```

Step 5: Open Liberty Monolith

Manually created server.xml

```
<server>
  <featureManager>

  <feature>microProfile-3.3</feature>
      <feature>jakartaee-8.0</feature>
      <feature>mpReactiveStreams-1.0</feature>
      <feature>mpReactiveMessaging-1.0</feature>
      <feature>monitor-1.0</feature>
  </featureManager>

  <library id="DB2Lib">
    <fileset dir="/opt/ol/wlp/usr/shared/resources/db2" includes="db2jcc4.jar db2jcc_license_cu.jar"/>
  </library>

  <dataSource id="OrderDS" jndiName="jdbc/orderds" type="javax.sql.XADataSource">
    <jdbcDriver libraryRef="DB2Lib"/>
    <properties.db2.jcc databaseName="${env.DB2_DBNAME}" password="${env.DB2_PASSWORD}" portNumber="${env.DB2_PORT}" serverName="${env.DB2_HOST}" user="${env.DB2_USER}"/>
    <connectionManager agedTimeout="0" connectionTimeout="180" maxIdleTime="1800" maxPoolSize="10" minPoolSize="1" reapTime="180"/>
  </dataSource>

  <httpEndpoint host="*" httpPort="9080" httpsPort="9443" id="defaultHttpEndpoint">
    <tcpOptions soReuseAddr="true"/>
  </httpEndpoint>

  <keyStore id="defaultKeyStore" password="whodunit"/>

  <applicationMonitor updateTrigger="mbean"/>

  <application id="customerOrderServicesApp" name="CustomerOrderServicesApp-0.1.0-SNAPSHOT.ear" type="ear" location="CustomerOrderServicesApp-0.1.0-SNAPSHOT.ear">
    <classloader apiTypeVisibility="spec, ibm-api, third-party" />
  </application>

</server>
```

Step 5: Open Liberty Monolith

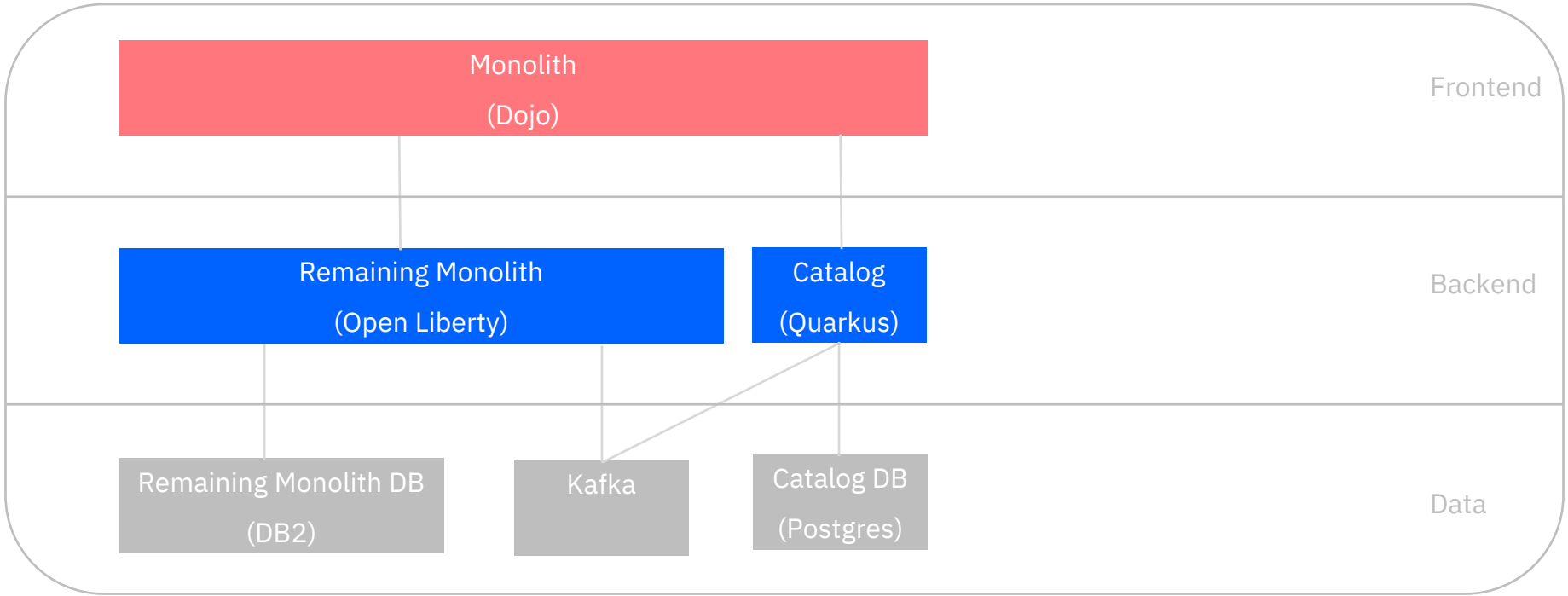
To be done:
Document JPA issues

Step 6: Strangled Catalog Service and Open Liberty Monolith

Description and Value	How did we get here?	Used Technologies	Containers
Strangled catalog service - Quarkus - Kafka for CQRS - Panache Open Liberty monolith - Modern project structure - No EJBs, but CDI - Kafka for CQRS -> Separate scalability	Inspired by Mono2Micro Manually	Quarkus OpenJ9 Kafka Open Liberty MicroProfile	Backend monolith (Liberty) Catalog service Frontend monolith Db2 Kafka Postgres

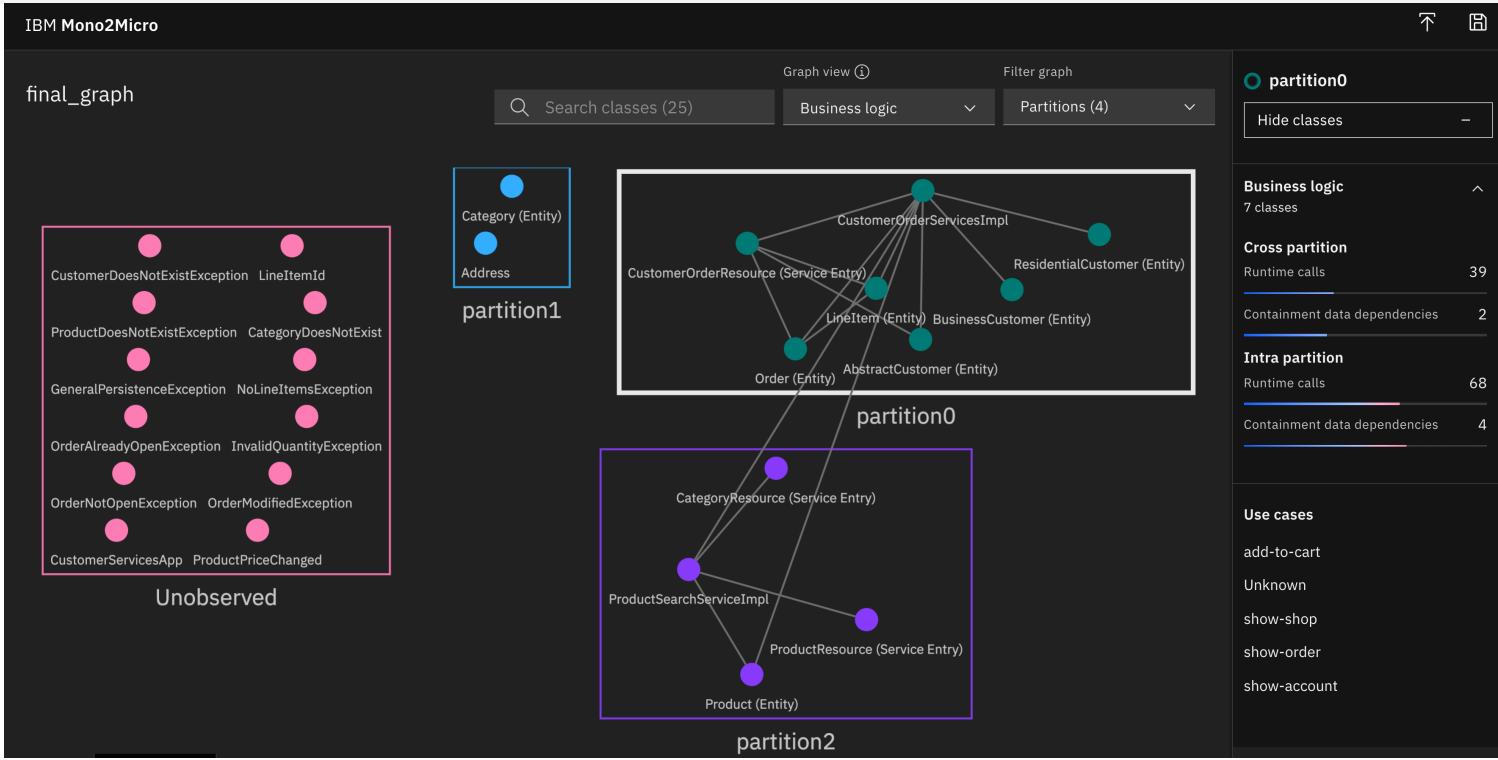
https://github.com/nheidloff/application-modernization-javaee-quarkus

Step 6: Strangled Catalog Service and Open Liberty Monolith

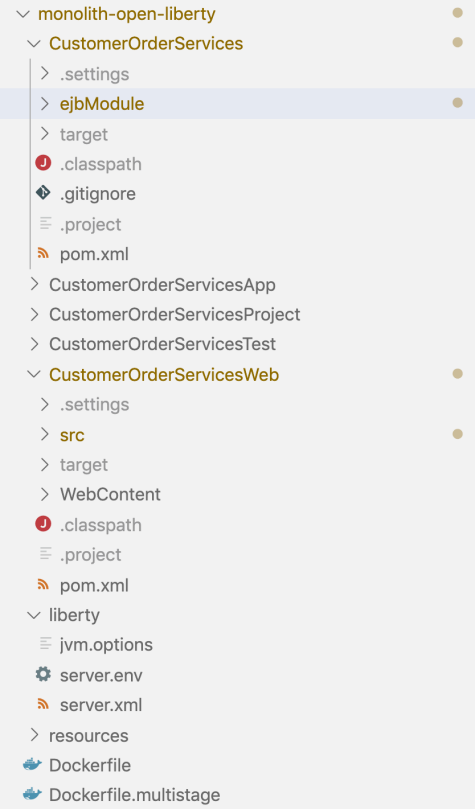


Step 6: Strangled Catalog Service and Open Liberty Monolith

Inspired by Mono2Micro



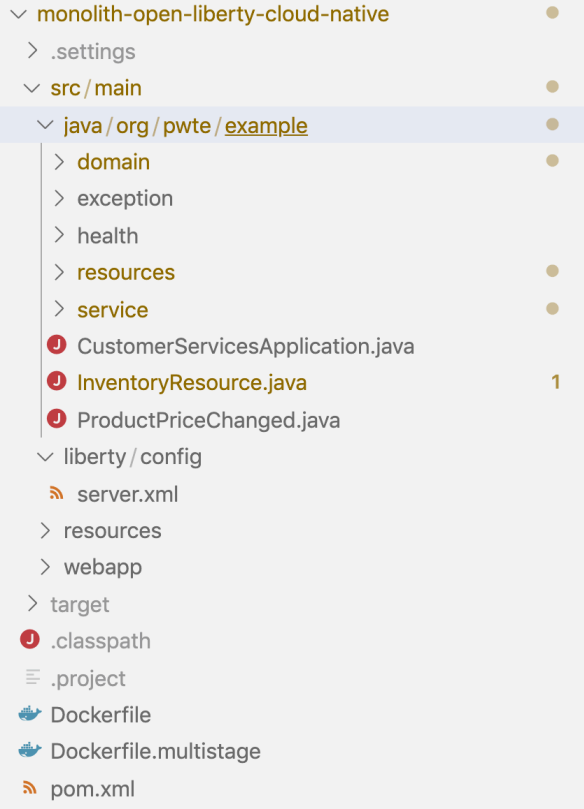
Step 6: Strangled Catalog Service and Open Liberty Monolith



Remaining Open Liberty Monolith - new project structure:

Only one pom and one project

-> To enable Open Liberty dev tools



Step 6: Strangled Catalog Service and Open Liberty Monolith

Remaining Open Liberty Monolith

To be done:
Document how to move from EJBs to CDI

Step 6: Strangled Catalog Service and Open Liberty Monolith

Remaining Open Liberty Monolith

To be done:
Document how to change JPA

Step 6: Strangled Catalog Service and Open Liberty Monolith

Remaining Open Liberty Monolith

To be done:
Add health checks, logging and monitoring

Step 6: Strangled Catalog Service and Open Liberty Monolith

Catalog service:

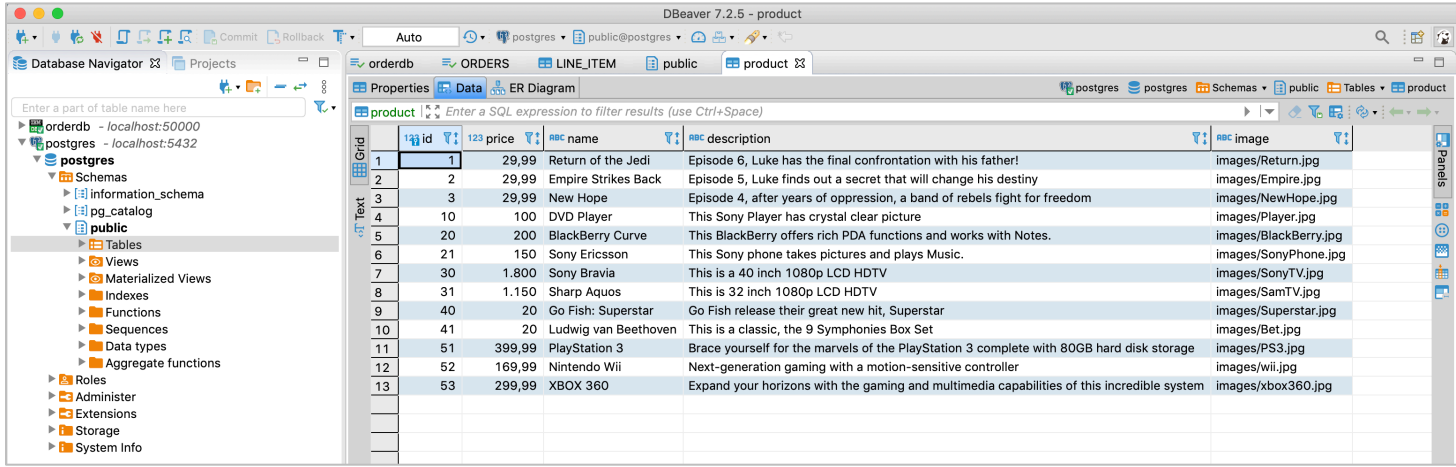
Panache

```
package com.ibm.catalog;

import java.math.BigDecimal;…

@Entity
@Cacheable
@NamedNativeQuery(name="product.by.cat.or.sub",
        query="select distinct p.id,p.name,p.price,p.description,p.image from product as p,productcategory as pc, category as c where (c.id = ? or c.parent = ?) AND pc.cat
        resultClass=Product.class)
public class Product extends PanacheEntity {

    public Product() {}

    public BigDecimal price;

    public String name;

    public String description;

    public String image;

    @ManyToMany(fetch=FetchType.EAGER)
    @JoinTable(name="productcategory",joinColumns={@JoinColumn(name="productid")},inverseJoinColumns={@JoinColumn(name="categoryid")})
    public Collection<Category> categories;

    @JsonbTransient
    public Collection<Category> getCategories() {
        return categories;
    }

    @JsonbTransient
    public void setCategories(Collection<Category> categories) {
        this.categories = categories;
    }
}
```

Step 6: Strangled Catalog Service and Open Liberty Monolith

Catalog service:

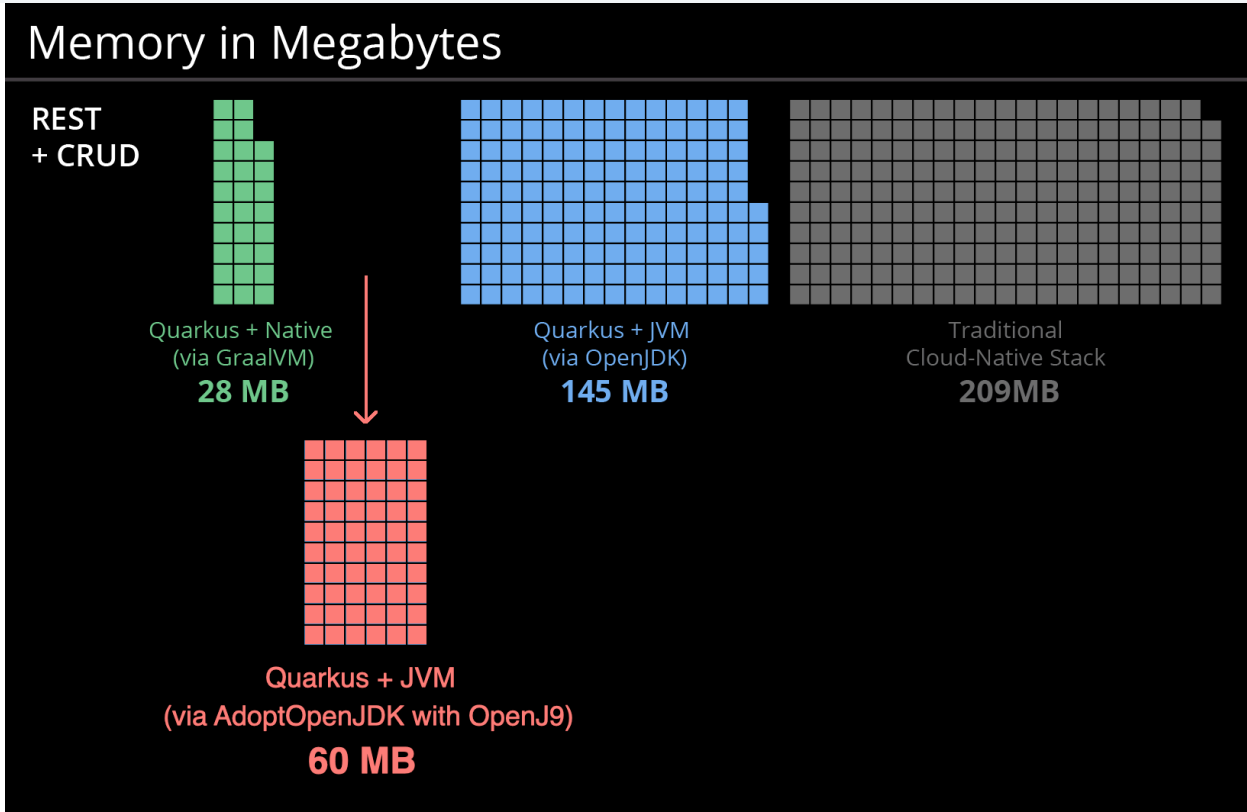
Postgres



Step 6: Strangled Catalog Service and Open Liberty Monolith

Quarkus with OpenJ9

To be done: Update, not only for initial memory usage

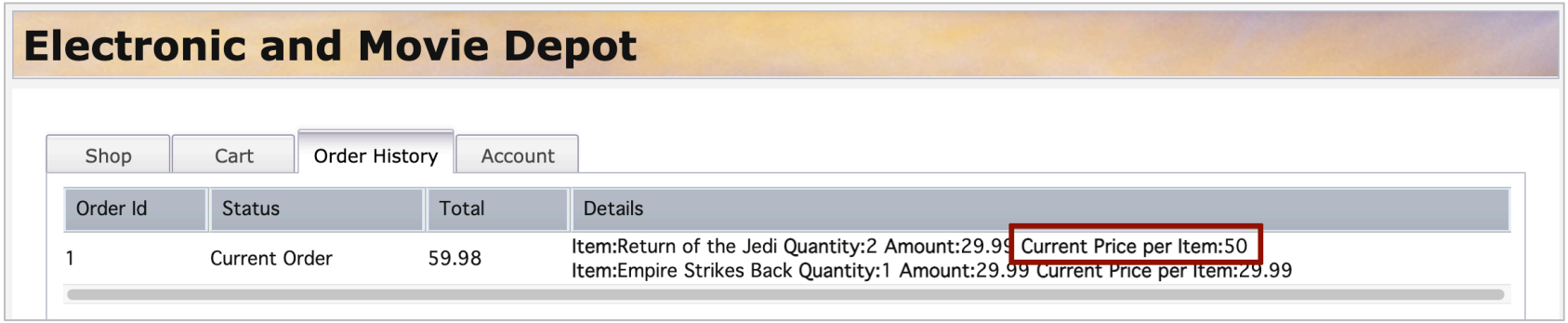


Step 6: Strangled Catalog Service and Open Liberty Monolith

Updated price from catalog service needs to be known in remaining monolith

-> CQRS (Command Query Responsibility Segregation)

```
Niklass-MBP-2:application-modernization-javaee-quarkus nheidloff$ curl -X PUT "http://localhost/CustomerOrderServicesWeb/jaxrs/Product/1"
 -H "accept: application/json" -H "Content-Type: application/json" -d "{\"id\":1, \"price\":50}"
```



Step 6: Strangled Catalog Service and Open Liberty Monolith

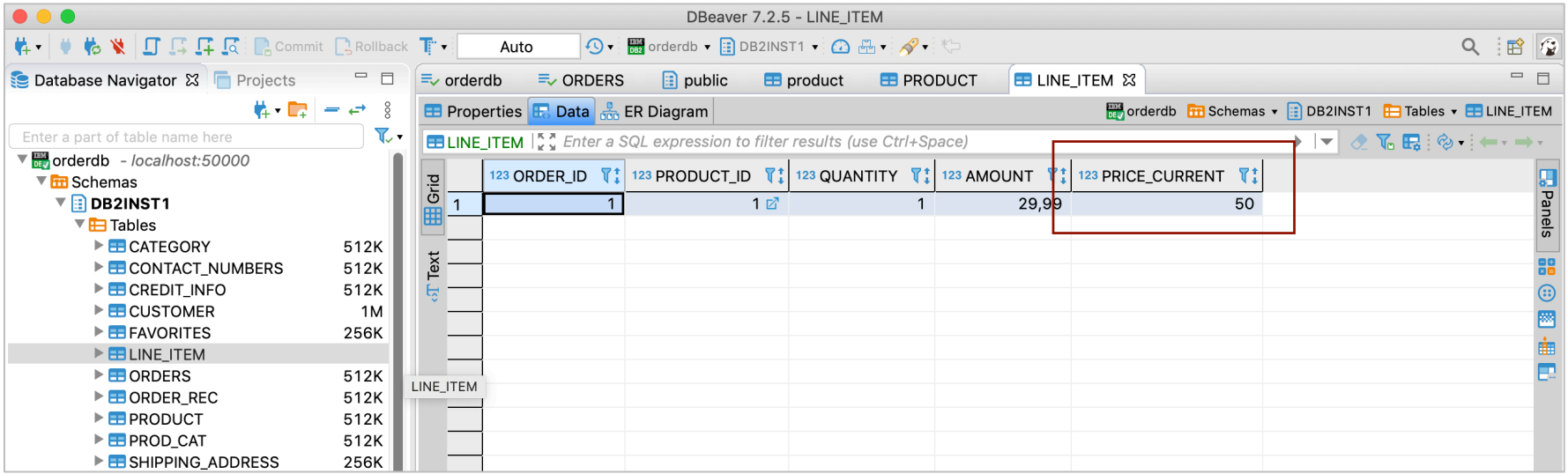
Catalog service: Send event when prices change

```
@PUT
@Consumes("application/json")
@Produces("application/json")
@Path("{id}")
@Transactional
public Product update(@PathParam("id
    System.out.println("/CustomerOrd

    Product existingProduct = entity
    if (existingProduct == null) {
        throw new WebApplicationExce
    }
    existingProduct.price = updatedP

    entityManager.persist(existingPr

    sendMessageToKafka(existingProdu

    return existingProduct;
}
```

```
private KafkaProducer<String, String> producer;

@PostConstruct
void initKafkaClient() {
    Map<String, String> config = new HashMap<>();
    config.put("bootstrap.servers", kafkaBootstrapServer);
    config.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
    config.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
    System.out.println("bootstrapping Kafka with config: " + config);

    producer = KafkaProducer.create(vertx, config);
}

public void sendMessageToKafka(Long productId, BigDecimal price) {
    String productIdString = productId.toString();
    String priceString = price.toString();
    try {
        KafkaProducerRecord<String, String> record = KafkaProducerRecord.create("product-price-updated", productIdString + "#" + priceString);
        producer.write(record, done -> System.out.println("Kafka message sent: product-price-updated - " + productIdString + "#" + priceString));
    } catch (Exception e) {
    }
}
```

Step 6: Strangled Catalog Service and Open Liberty Monolith

Remaining monolith: New price is cached in Db2 database



Step 6: Strangled Catalog Service and Open Liberty Monolith

Remaining monolith: New price is cached in Db2 database

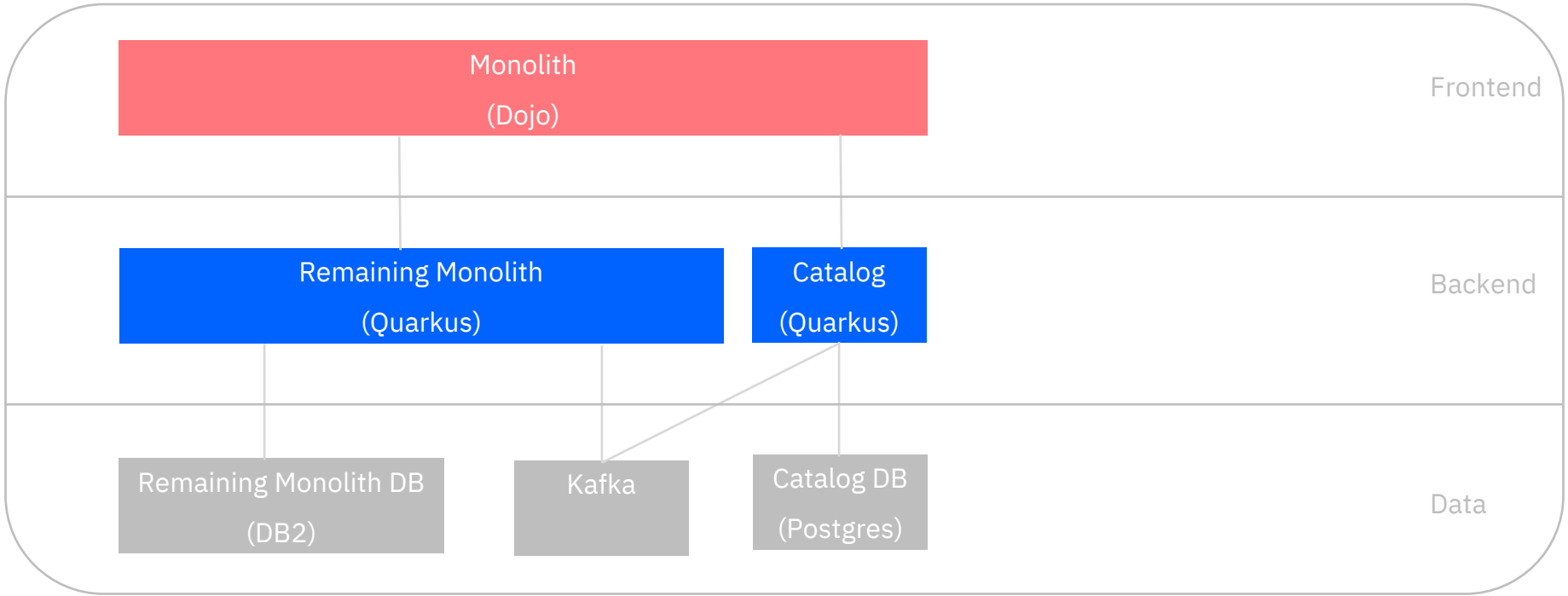
```
@ApplicationScoped
public class ProductPriceChanged {

    @Inject
    CustomerOrderServicesImpl customerOrderServices;

    @Incoming("product-price-updated")
    public String process(String message) {
        System.out.println("Open Liberty - Kafka message received: product-price-updated - " + message);
        String productId = "";
        String newPrice = "0";
        try {
            productId = message.substring(0, message.indexOf("#"));
            newPrice = message.substring(message.indexOf("#") + 1, message.length());
            customerOrderServices.updateLineItem(productId, newPrice);
        }
        catch (Exception e) {}
        return message;
    }
}
```

Step 7: Strangled Catalog Service and Quarkus Monolith

Description and Value	How did we get here?	Used Technologies	Containers
Strangled catalog service - Reactive endpoints - Reactive Postgres access -> More efficient than synchronous code Quarkus monolith - Compiled to native binary -> Faster startup time and less memory usage	Manually	Quarkus OpenJ9 Kafka MicroProfile	Backend monolith (Quarkus) Catalog service Frontend monolith Db2 Kafka Postgres

https://github.com/nheidloff/application-modernization-javaee-quarkus

Step 7: Strangled Catalog Service and Quarkus Monolith



Step 7: Strangled Catalog Service and Quarkus Monolith

Catalog service:

Reactive endpoints

```
@Path("/CustomerOrderServicesWeb/jaxrs/Product")
@ApplicationScoped
@Produces("application/json")
public class ProductResource {

    @Inject
    io.vertx.axle.pgclient.PgPool client;
    private static int MAXIMAL_DURATION = 5000;

    @GET
    public CompletionStage<List<Product>> getProductsByCategory(@QueryParam(value="categoryId") int categoryId) {
        System.out.println("/CustomerOrderServicesWeb/jaxrs/Product invoked in Quarkus reactive catalog service");

        CompletionStage<List<Product>> products = getProducts();
        CompletionStage<List<ProductCategory>> productCategories = getProductCategories();
        Long categoryIdLong = new Long(categoryId);

        return products
            .thenCombine(productCategories, (productsRead, productCategoriesRead) -> {
                List<Product> output = new ArrayList<Product>();
                List<ProductCategory> productCategoriesFiltered = productCategoriesRead.stream().filter(productCategoryRead -> {
                    return productCategoryRead.categoryid.equals(categoryIdLong);
                })
                .collect(Collectors.toList());

                productCategoriesFiltered.forEach(productCategoryFiltered -> {
                    productsRead.forEach(productRead -> {
                        if (productRead.id.equals(productCategoryFiltered.productid)) {
                            output.add(productRead);
                        }
                    });
                });
                return output;
            });
    }
```

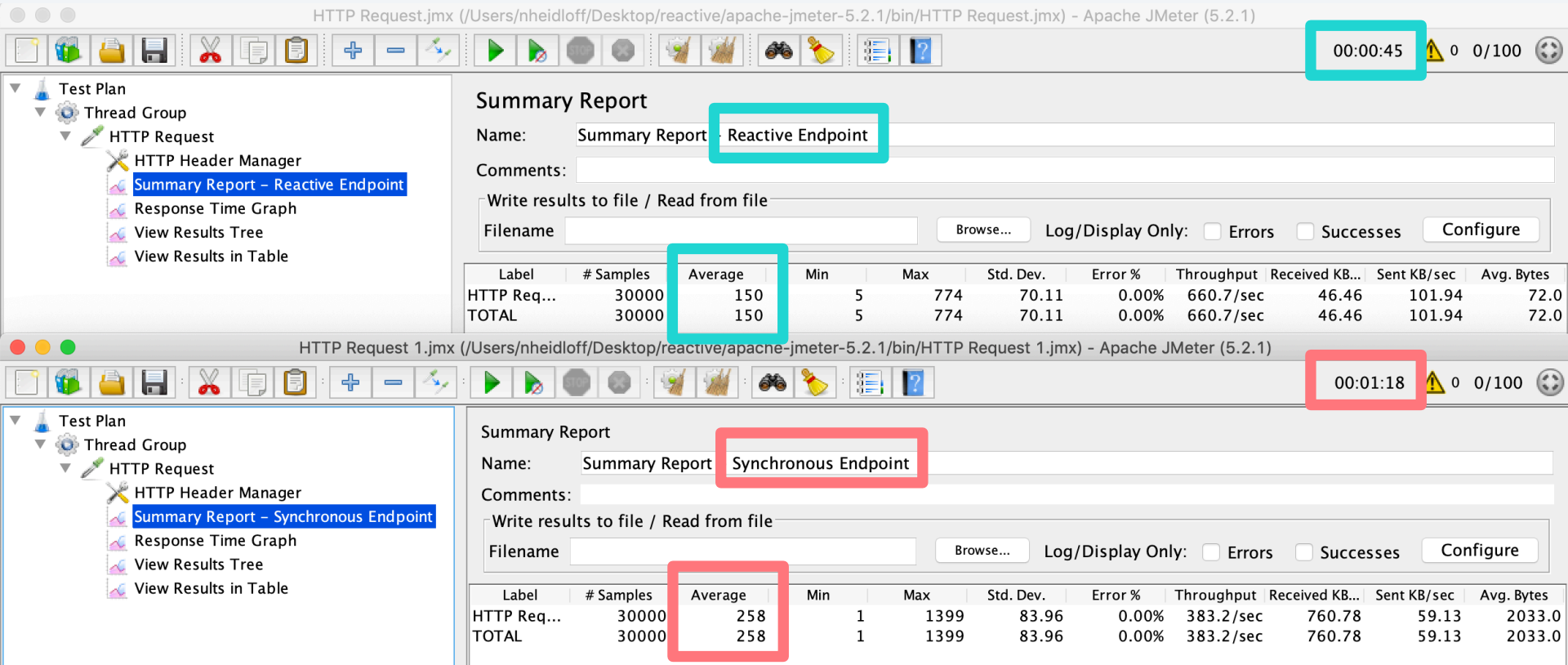
Step 7: Strangled Catalog Service and Quarkus Monolith

Catalog service:

Reactive Postgres access

```
public CompletionStage<List<Product>> getProducts() {
    String statement = "SELECT id, price, name, description, image FROM product";
    return client.preparedQuery(statement)
            .toCompletableFuture()
            .orTimeout(MAXIMAL_DURATION, TimeUnit.MILLISECONDS)
            .exceptionally(throwable -> {
                System.out.println(throwable);
                return null;
            })
            .thenApply(rows -> {
                List<Product> products = new ArrayList<>(rows.size());
                rows.forEach(row -> products.add(fromRow(row)));
                return products;
            });
}
```

Step 7: Strangled Catalog Service and Quarkus Monolith



Step 7: Strangled Catalog Service and Quarkus Monolith

Remaining monolith (Quarkus)

Compiled to native code

```
## Stage 1 : build with maven builder image with native capabilities
FROM quay.io/quarkus/centos-quarkus-maven:20.2.0-java11 AS build
COPY pom.xml /usr/src/app/
RUN mvn -f /usr/src/app/pom.xml -B de.qaware.maven:go-offline-maven-plugin:1.2.5:resolve-dependencies
COPY src /usr/src/app/src
USER root
RUN chown -R quarkus /usr/src/app
USER quarkus
RUN mvn -f /usr/src/app/pom.xml -Pnative clean package

## Stage 2 : create the docker final image
FROM registry.access.redhat.com/ubi8/ubi-minimal
WORKDIR /work/
COPY --from=build /usr/src/app/target/*-runner /work/application

# set up permissions for user `1001`
RUN chmod 775 /work /work/application \
  && chown -R 1001 /work \
  && chmod -R "g+rwX" /work \
  && chown -R 1001:root /work

EXPOSE 8080
USER 1001

CMD ["./application", "-Dquarkus.http.host=0.0.0.0"]
```

Step 7: Strangled Catalog Service and Quarkus Monolith

Remaining monolith (Quarkus)

Quarkus doesn’t allow blocking code on main thread

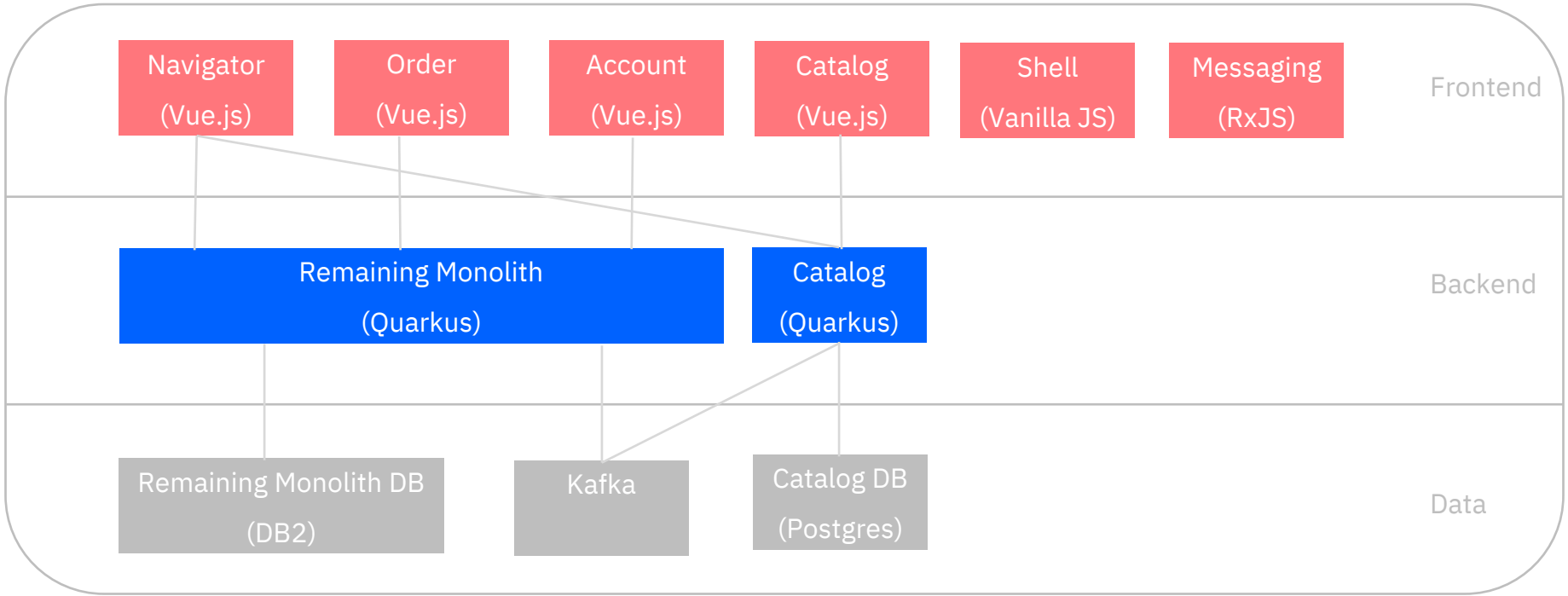
```
@ApplicationScoped
public class ProductPriceChanged {

    @PersistenceContext
    protected EntityManager em;

    @Inject
    CustomerOrderServicesImpl customerOrderServices;

    @Transactional
    @Incoming("product-price-updated")
    public CompletionStage<?> process(String message) {
        System.out.println("Quarkus - Kafka message received: product-price-updated - " + message);
        try {
            ManagedExecutor executor = ManagedExecutor.builder()
                .maxAsync(10)
                .propagated(ThreadContext.CDI,
                        ThreadContext.TRANSACTION)
                .build();

            ThreadContext threadContext = ThreadContext.builder()
                    .propagated(ThreadContext.CDI,
                            ThreadContext.TRANSACTION)
                    .build();

            return executor.runAsync(threadContext.contextualRunnable(() -> {
                customerOrderServices.updateLineItem(message);
            }));
        }
        catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

Step 8: Quarkus Backend and Micro Frontends

Description and Value	How did we get here?	Used Technologies	Containers
Break down frontend monolith in micro frontends -> Independent updates to separate pages	Manually	single-spa Vue.js RxJS	Backend monolith (Quarkus) Catalog service Frontend monolith Db2 Kafka Postgres Navigator, catalog, account, order, shell, messaging

https://github.com/nheidloff/application-modernization-javaee-quarkus

Step 8: Quarkus Backend and Micro Frontends



Step 6: Strangled Catalog Service and Open Liberty Monolith

To be done:

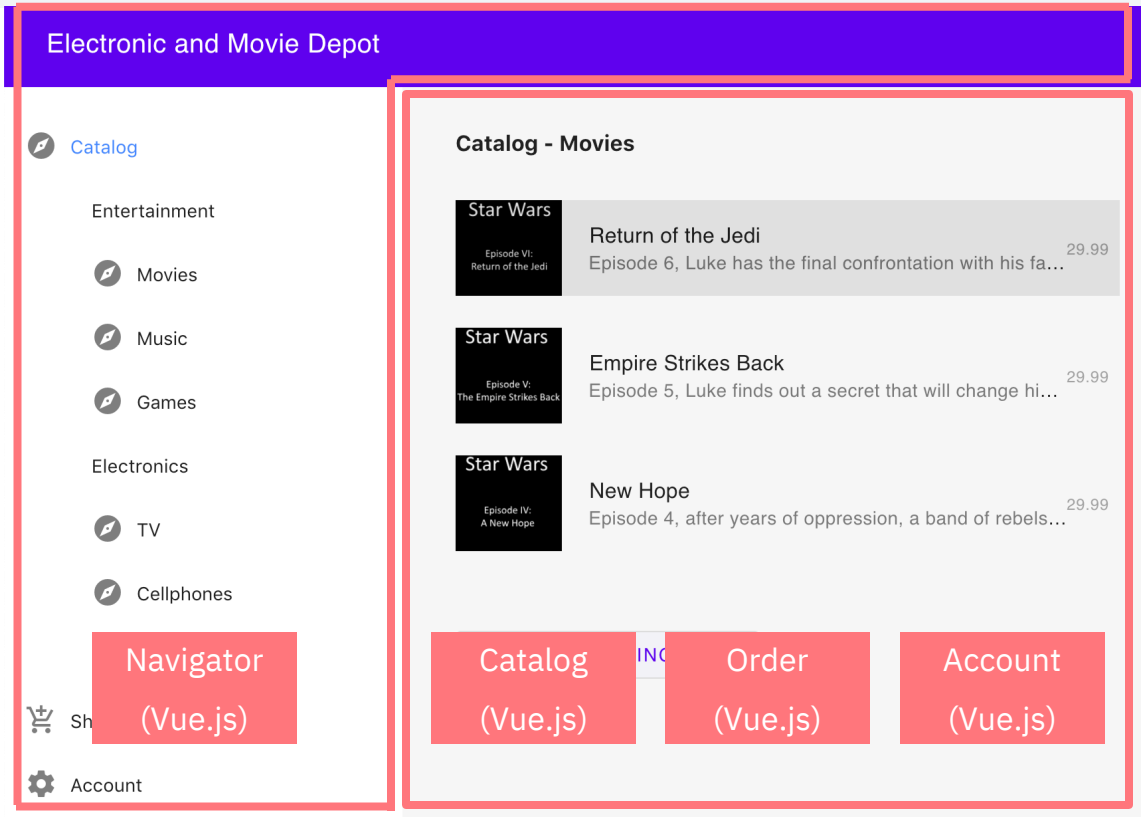
Add use case – Add ratings to products

- **Change database**
- **Change backend code**
- **Change frontend code**

Micro Frontends via single-spa

Shell
(Vanilla JS)

Messaging
(RxJS)



Micro Frontends via single-spa

Navigator
(Vue.js)

Catalog
(Vue.js)

Order
(Vue.js)

Account
(Vue.js)

Shell
(Vanilla JS)

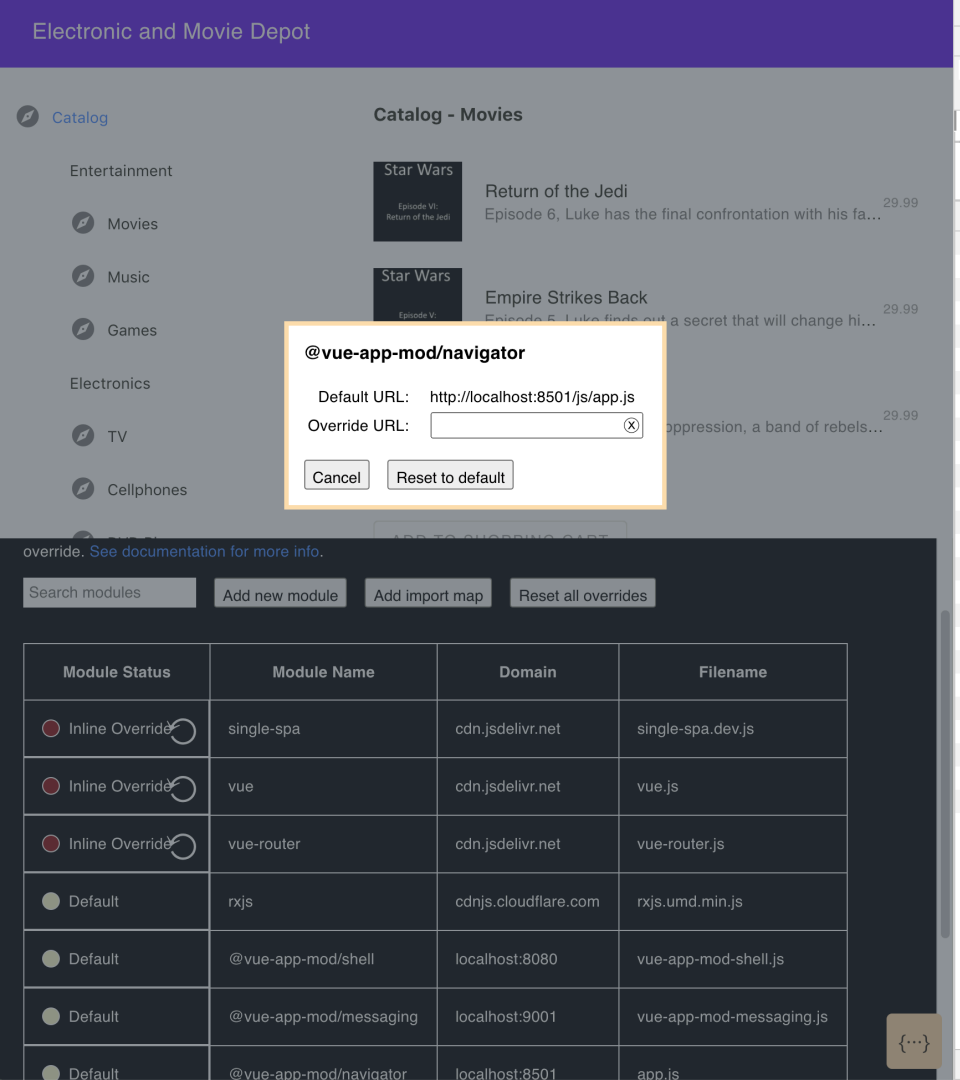
```
import { registerApplication, start } from "single-spa";

registerApplication({
  name: "@vue-app-mod/navigator",
  app: () => System.import("@vue-app-mod/navigator"),
  activeWhen: ["/"],
});

registerApplication({
  name: "@vue-app-mod/order",
  app: () => System.import("@vue-app-mod/order"),
  activeWhen: ["/", "/catalog", "/order"],
});

registerApplication({
  name: "@vue-app-mod/account",
  app: () => System.import("@vue-app-mod/account"),
  activeWhen: "/account",
});

registerApplication({
  name: "@vue-app-mod/catalog",
  app: () => System.import("@vue-app-mod/catalog"),
  activeWhen: ["/", "/catalog"],
});

start();
```

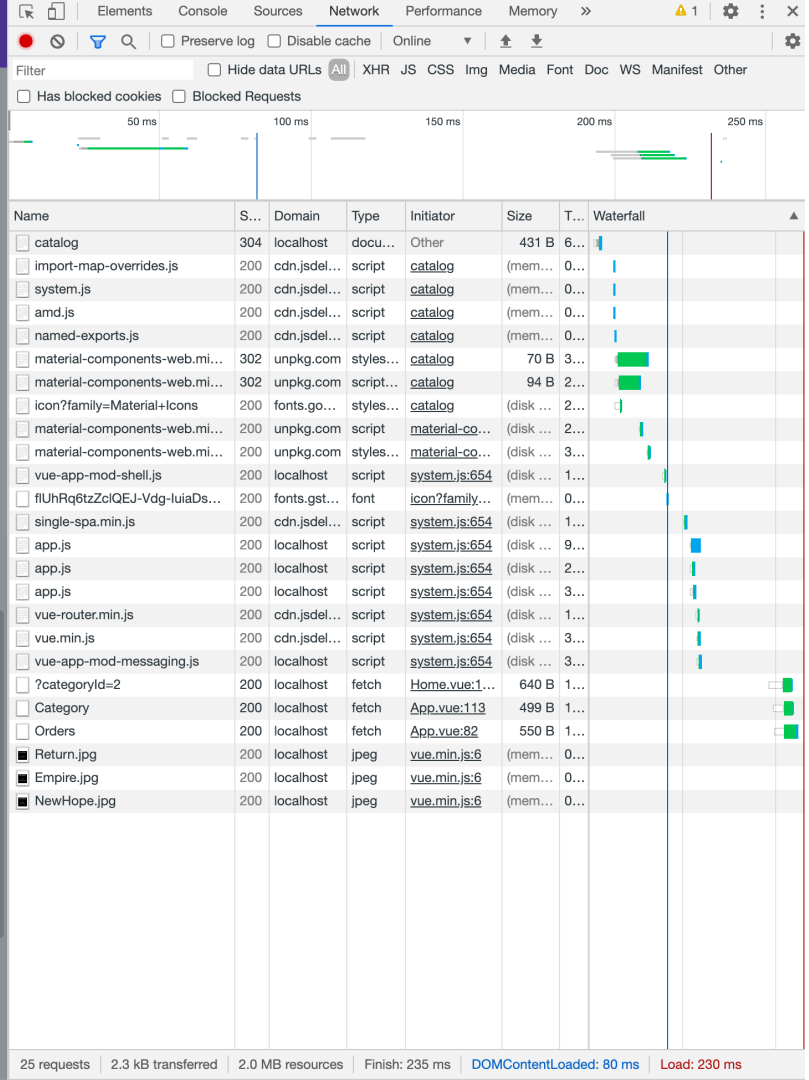
Messaging
(RxJS)

```
export default {
  ...

  getObservable(microFrontendName) {
    switch (microFrontendName) {
      case MICRO_FRONTEND_NAVIGATOR:
        if (!observableForNavigator) {
          observableForNavigator = new Subject()
        }
        return observableForNavigator
      ...
    }
  },

  send(message) {
    let topic = message.topic
    let commandId = message.commandId

    switch (topic) {
      case TOPIC_COMMAND_ADD_ITEM:
        if (observableForOrder) {
          observableForOrder.next(message)
        }
        break
      ...
```

Micro Frontends via single-spa

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta name="importmap-type" content="systemjs-importmap" />
  <script type="systemjs-importmap">
    {
      "imports": {
        "single-spa": "https://cdn.jsdelivr.net/npm/single-spa@5.5.1/lib/system/single-spa.min.js",
        "vue": "https://cdn.jsdelivr.net/npm/vue@2.6.11/dist/vue.min.js",
        "vue-router": "https://cdn.jsdelivr.net/npm/vue-router@3.1.6/dist/vue-router.min.js",
        "rxjs": "http://cdnjs.cloudflare.com/ajax/libs/rxjs/6.6.3/rxjs.umd.min.js",

        "@vue-app-mod/shell": "http://localhost:8080/vue-app-mod-shell.js",
        "@vue-app-mod/messaging": "http://localhost:9001/vue-app-mod-messaging.js",
        "@vue-app-mod/navigator": "http://localhost:8501/js/app.js",
        "@vue-app-mod/order": "http://localhost:8504/js/app.js",
        "@vue-app-mod/catalog": "http://localhost:8503/js/app.js",
        "@vue-app-mod/account": "http://localhost:8502/js/app.js"
      }
    }
  </script>
  <script src="https://cdn.jsdelivr.net/npm/import-map-overrides/dist/import-map-overrides.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/systemjs/dist/system.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/systemjs/dist/extras/amd.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/systemjs/dist/extras/named-exports.js"></script>
  <link href="https://unpkg.com/material-components-web@latest/dist/material-components-web.min.css" rel="stylesheet">
  <script src="https://unpkg.com/material-components-web@latest/dist/material-components-web.min.js"></script>
  <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
</head>
<body>
  <div style="display: flex;flex-direction: row;">
    <div id="navigator" style="order: 1;flex-grow: 2;max-width: 300px;width:300px"></div>
    <div id="catalog" style="order: 2; flex-grow: 10;left: 320px;position: fixed;"></div>
    <div id="order" style="order: 2; flex-grow: 10;left: 320px;position: fixed;"></div>
    <div id="account" style="order: 2; flex-grow: 10;left: 320px;position: fixed;"></div>
  </div>
  <script>
    System.import('@vue-app-mod/shell');
  </script>
  <import-map-overrides-full show-when-local-storage="devtools" dev-libs></import-map-overrides-full>
</body>
</html>
```


Electronic and Movie Depot
Catalog
Entertainment
Movies
Music
Games
Electronics
TV
Cellphones
Catalog - Movies
Star Wars
Episode VI: Return of the Jedi
Return of the Jedi
Episode 6, Luke has the final confrontation with his fa...
29.99
Star Wars
Episode V:
Empire Strikes Back
29.99
@vue-app-mod/navigator
Default URL: http://localhost:8501/js/app.js
Override URL:
Cancel
Reset to default
29.99
override. See documentation for more info.
Search modules
Add new module
Add import map
Reset all overrides
Module Status
Module Name
Domain
Filename
Inline Override
single-spa
cdn.jsdelivr.net
single-spa.dev.js
Inline Override
vue
cdn.jsdelivr.net
vue.js
Inline Override
vue-router
cdn.jsdelivr.net
vue-router.js
Default
rxjs
cdnjs.cloudflare.com
rxjs.umd.min.js
Default
@vue-app-mod/shell
localhost:8080
vue-app-mod-shell.js
Default
@vue-app-mod/messaging
localhost:9001
vue-app-mod-messaging.js
Default
@vue-app-mod/navigator
localhost:8501
app.js


Elements
Console
Sources
Network
Performance
Memory
Preserve log
Disable cache
Online
Filter
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
Has blocked cookies
Blocked Requests
50 ms
100 ms
150 ms
200 ms
250 ms
Name
S...
Domain
Type
Initiator
Size
T...
Waterfall
catalog 304 localhost docu... Other 431 B 6...
import-map-overrides.js 200 cdn.jsdel... script catalog (mem... 0...
system.js 200 cdn.jsdel... script catalog (mem... 0...
amd.js 200 cdn.jsdel... script catalog (mem... 0...
named-exports.js 200 cdn.jsdel... script catalog (mem... 0...
material-components-web.mi... 302 unpkg.com styles... catalog 70 B 3...
material-components-web.mi... 302 unpkg.com script... catalog 94 B 2...
icon?family=Material+Icons 200 fonts.go... styles... catalog (disk ... 2...
material-components-web.mi... 200 unpkg.com script material-co... (disk ... 2...
material-components-web.mi... 200 unpkg.com styles... material-co... (disk ... 3...
vue-app-mod-shell.js 200 localhost script system.js:654 (disk ... 1...
flUhRq6tzZclQEJ-Vdg-IuiaDs... 200 fonts.gst... font icon?family... (mem... 0...
single-spa.min.js 200 cdn.jsdel... script system.js:654 (disk ... 1...
app.js 200 localhost script system.js:654 (disk ... 9...
app.js 200 localhost script system.js:654 (disk ... 2...
app.js 200 localhost script system.js:654 (disk ... 3...
vue-router.min.js 200 cdn.jsdel... script system.js:654 (disk ... 1...
vue.min.js 200 cdn.jsdel... script system.js:654 (disk ... 3...
vue-app-mod-messaging.js 200 localhost script system.js:654 (disk ... 3...
?categoryId=2 200 localhost fetch Home.vue:1... 640 B 1...
Category 200 localhost fetch App.vue:113 499 B 1...
Orders 200 localhost fetch App.vue:82 550 B 1...
Return.jpg 200 localhost jpeg vue.min.js:6 (mem... 0...
Empire.jpg 200 localhost jpeg vue.min.js:6 (mem... 0...
NewHope.jpg 200 localhost jpeg vue.min.js:6 (mem... 0...
25 requests
2.3 kB transferred
2.0 MB resources
Finish: 235 ms
DOMContentLoaded: 80 ms
Load: 230 ms

Micro Frontends – Messaging via RxJS for loosely Coupling Command Pattern – Example: Add Item to Cart