Magento Live
Australia | 2018

Techinal Direction of Magento

© 2018 Magento, Inc.

Anton Kril

Director of Architecture, Magento

@antonkril

© 2018 Magento, Inc.

Tracks

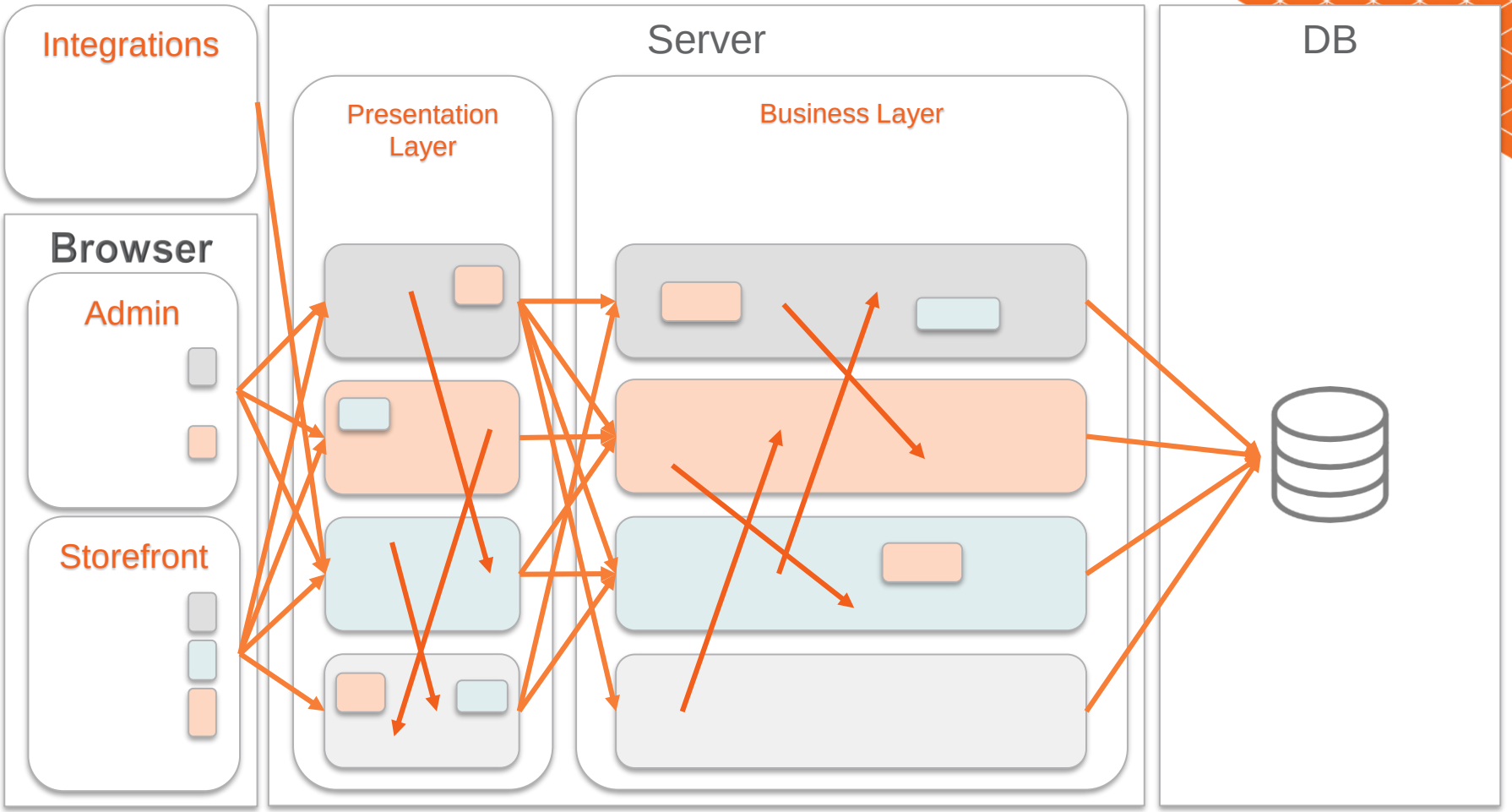
- PWA using React
- Separation of APIs for storefront and admin
- Command & Query APIs
- Decoupling & modularity
- Cloud native

© 2018 Magento, Inc.

Goals

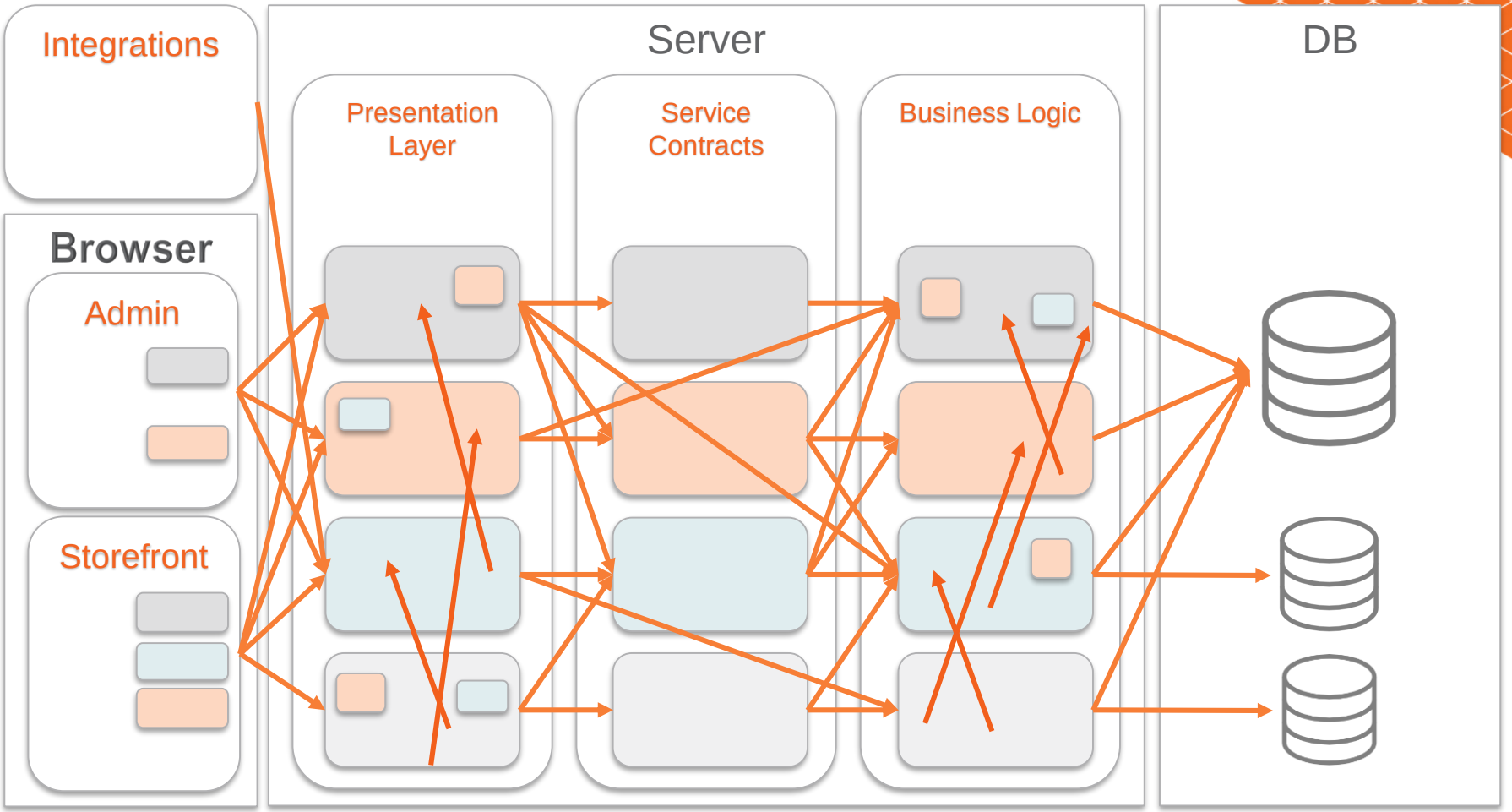
- Evolutionary architecture
- Improved dev experience
- Leaner core, moving behavior to cloud
- Performance & scalability

© 2018 Magento, Inc.



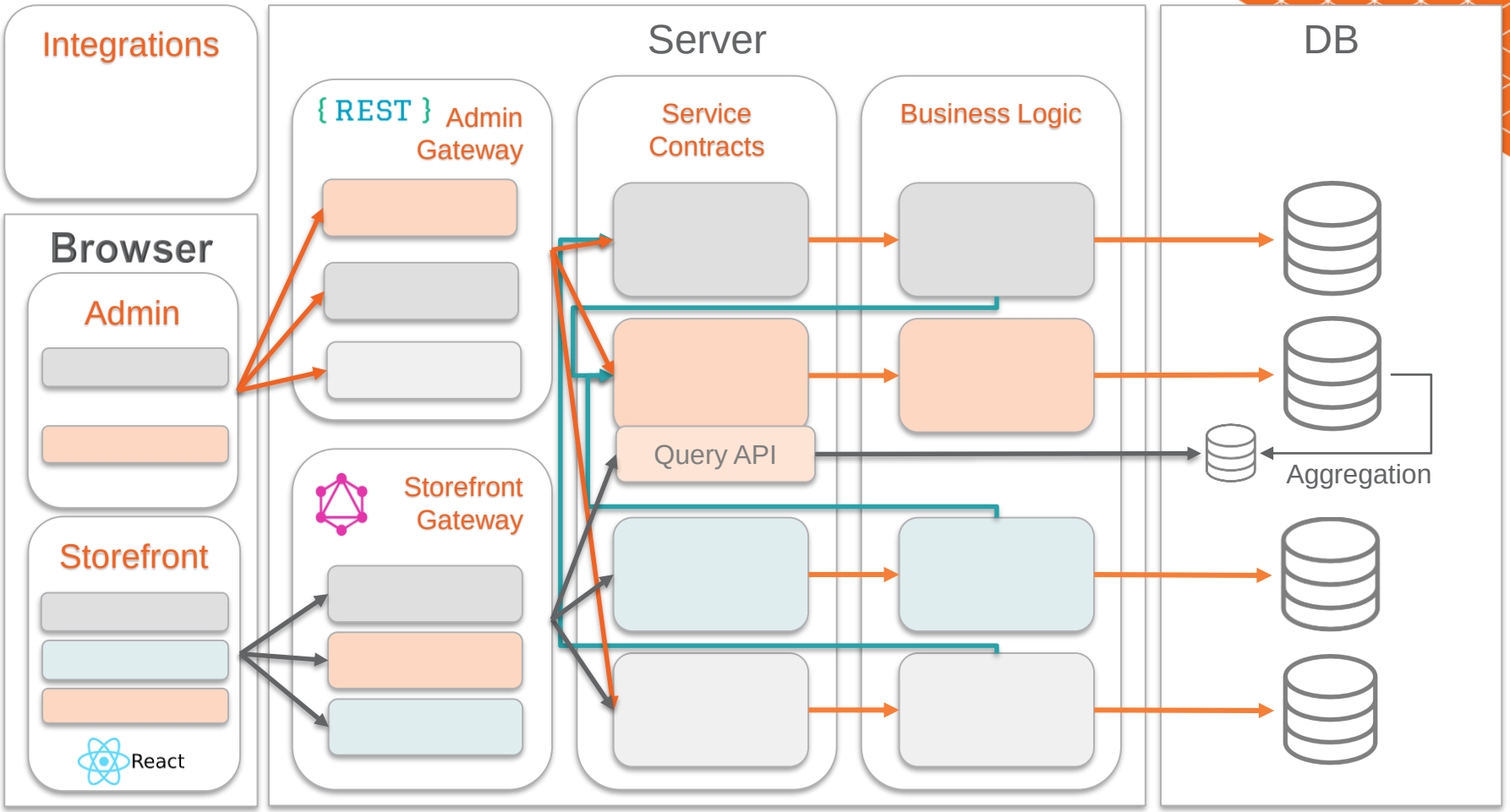
© 2018 Magento, Inc.

Catalog | Customers | Checkout | Order Management | . . .

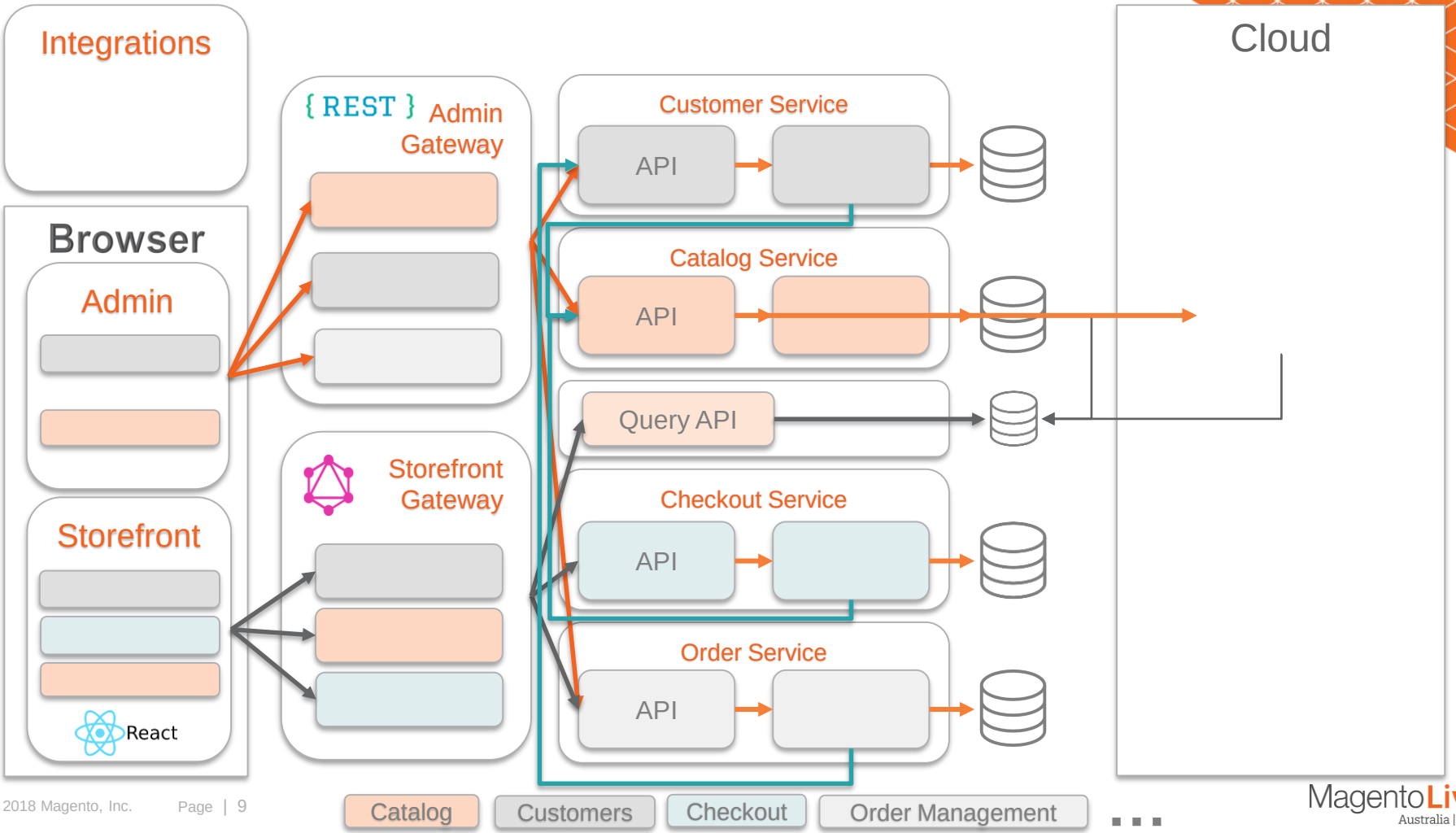


Catalog | Customers | Checkout | Order Management | . . .

© 2018 Magento, Inc.



Catalog | Customers | Checkout | Order Management | ...



© 2018 Magento, Inc.

Progressive Web App

- Service Worker
- React-based storefront frontend (Admin stays the same)
- Redux for state management
- Webpack-based build system
- Layout-like customizations & modularity

http://bit.ly/pwa-studio-introduction

© 2018 Magento, Inc.

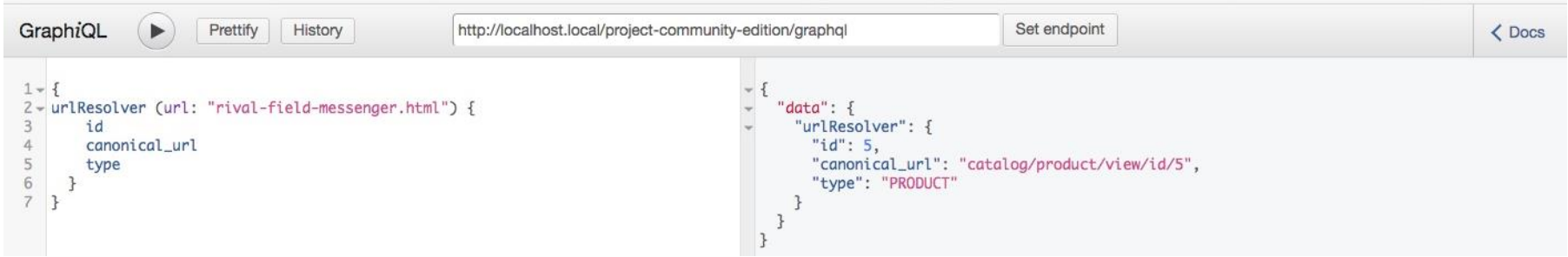
GraphQL

File: /vendor/magento/module-url-rewrite-graph-ql/etc/graphql.xml

```
<config>
   <type xsi:type="OutputType" name="Query">
      <field xsi:type="ObjectOutputField" name="urlResolver" type="EntityUrl"
             resolver="Magento\UrlRewriteGraphQl\Model\Resolver\UrlRewrite">
         <argument xsi:type="ScalarArgument" name="url" type="String" required="true"/>
      </field>
   </type>
   <type xsi:type="OutputType" name="EntityUrl">
      <field xsi:type="ScalarOutputField" name="id" type="Int" description="..."/>
      <field xsi:type="ScalarOutputField" name="canonical_url" type="String" description="..."/>
      <field xsi:type="ObjectOutputField" name="type" type="UrlRewriteEntityTypeEnum" description="..."/>
   </type>
   <type xsi:type="Enum" name="UrlRewriteEntityTypeEnum"/>
</config>
```

More details in the next talk by Oleksii Korshenko

http://bit.ly/magento-graphql-preview-demo



Declarative DB Schema

```
<table name  "catalog_product_entity">
   <field name="entity_id">
      <type>integer</type>
      <unsigned>1</unsigned>
      <primary>1</primary>
      <identity>1</identity>
      <comment>Entity ID</comment>
   </field>
   <field name=”price">
      <type>integer</type>
      <unsigned>1</unsigned>
      <default>0</default>
   </field>
</table>
```

≠

```
<table name  "catalog_product_entity">
   <field name="entity_id">
      <type>integer</type>
      <unsigned>1</unsigned>
      <primary>1</primary>
      <identity>1</identity>
      <comment>Entity ID</comment>
   </field>
</table>
```

```
ALTER TABLE catalog_product_entity ADD COLUMN `price` int(11) unsigned DEFAULT NULL;
```

© 2018 Magento, Inc.

Resources

Technical guidelines

Edit this page on GitHub
Give us feedback

ON THIS PAGE

About this document
- Overview
- Text conventions
- Guidelines for Magento 2.2 only

1. Basic programming principles
2. Class design
3. Dependency injection
4. Interception
5. Exceptions
6. Application layers
   - 6.1. All layers
   - 6.2. Presentation layer
   - 6.3. Data Access (Persistence) layer
   - 6.4. Service Contracts (Application) layer
7. Configuration
8. Modularity
9. Browser-Server interaction in web application
10. JavaScript (JS) application
11. Testing

About this document

Overview

This document lists the fundamental coding and application design principles that guide Magento 2 developer team members.

Magento core developers use this document as a reference during code reviews; some rules have corresponding code checks in the Magento static tests.

These guidelines came from many years of hard work, experience, and discussions. We strongly believe that new technical initiatives should follow these recommendations, and the existing code should be improved to meet them.

Text conventions

Use RFC2119 to interpret keywords like:

- MUST and MUST NOT
- REQUIRED
- SHALL and SHALL NOT
- SHOULD and SHOULD NOT
- RECOMMENDED

http://bit.ly/2-2-technical-guidelines

© 2018 Magento, Inc.

Deprecated Code Example

```
/**
 * @deprecated since 2.1.0
 * @see \Magento\Framework\Model\ResourceModel\Db\AbstractDb::save()
 */
public function save()
{
    // ...
}
```

Version increase scenarios

This table lists code change scenarios and which version number it affects. Use this table to understand what changes Magento can make and which version number gets increased for that change.

API/Customization Point	Code Change	Version Change
PHP Interface (marked with `@api`)	New interface	MINOR
	New method added	MINOR
	Interface removed	MAJOR
	Method removed	MAJOR
	New required method argument	MAJOR
	Removed the last argument for a method	MINOR
	Changed a method signature (excluding last argument removal)	MAJOR
PHP Class (marked with `@api`)	New Class	MINOR
	New method added	MINOR
	Class removed	MAJOR
	Method removed	MAJOR
	New required method argument	MAJOR
	Removed a non-last argument	MAJOR
	New required constructor object argument	MINOR
	New required constructor scalar	MAJOR

http://bit.ly/magento-versioning

© 2018 Magento, Inc.

The Twelve Factors

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

IV. Backing services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

VI. Processes
Execute the app as one or more stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity
Keep development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin/management tasks as one-off processes

https://12factor.net/

© 2018 Magento, Inc.

© 2018 Magento, Inc.

Magento 2 Web API with GraphQL

© 2018 Magento, Inc.

Oleksii Korshenko

Magento 2 Architect, Community Engineering

© 2018 Magento, Inc.

Magento Web API

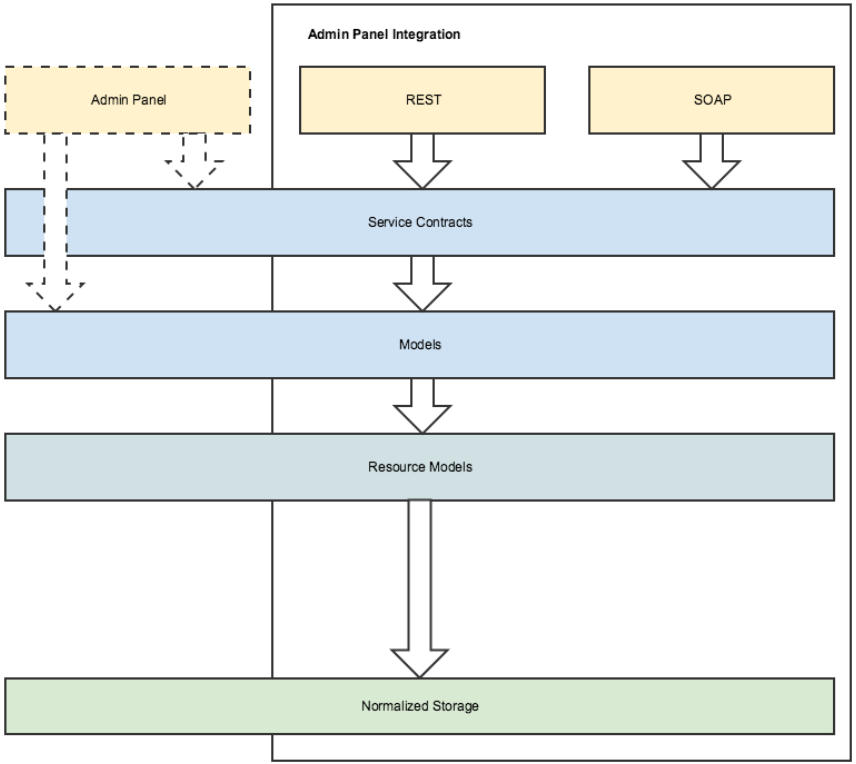
Basic Components

Current Web API Overview

- Headless Magento, Frontend integrations
  - ✓ Advanced frontend (PWA)
- Backend Integrations which Magento uses to do business
  - ✓ Shipping
  - ✓ Order management
- Backend Integrations which use Magento data
  - ✓ Analytics
  - ✓ Product Management

© 2018 Magento, Inc.

Web API – Request Processing – High Level Overview



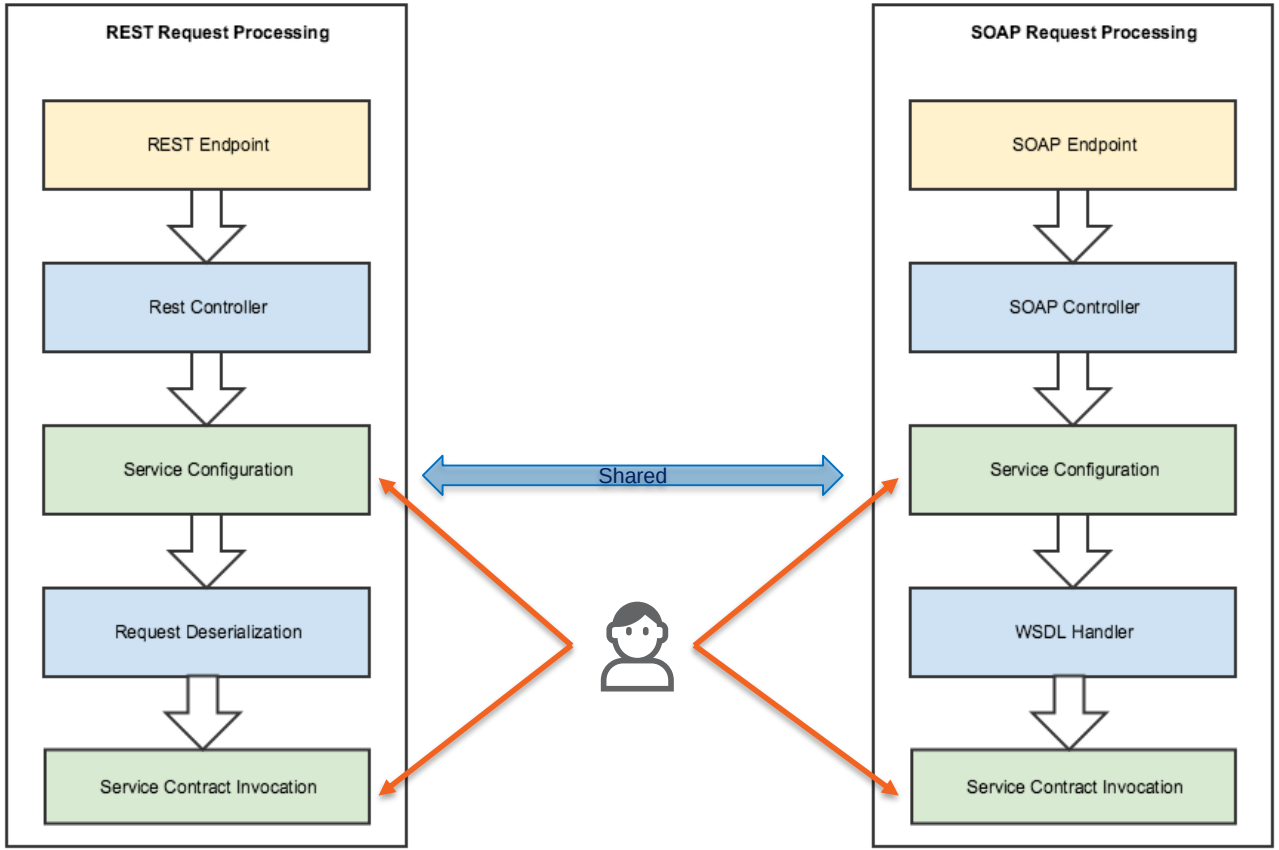
© 2018 Magento, Inc.

Endpoints

- /rest
  - Handles all REST requests
  - Supports JSON/XML
- /soap
  - Used as entry points for SOAP requests
  - Schema defined in WSDL
- /schema
  - Schema generated in Swagger
  - Schema generated in WSDL

© 2018 Magento, Inc.

Web API framework



© 2018 Magento, Inc.

Configuration and Routing

```
<!-- Product Service -->
<route url="/V1/products" method="POST">
  <service class="Magento\Catalog\Api\ProductRepositoryInterface" method="save"/>
 <resources>
   <resource ref="Magento_Catalog::products" />
 </resources>
</route>
```

© 2018 Magento, Inc.

REST

- ✅ POST should be used for entity(s) creation
- ✅ PUT should be used for entity(s) update
- ✅ PATCH should be used for patch entity(s) update (when unchanged fields are omitted in the request)
- ✅ Must return responses with standard HTTP status codes
- ✅ Designed for system integrations, mobile app, one-page apps
- ✅ Supports tokens and cookies authentication
- ✅ OAuth 1.0 with token exchange
- ✅ Anonymous access

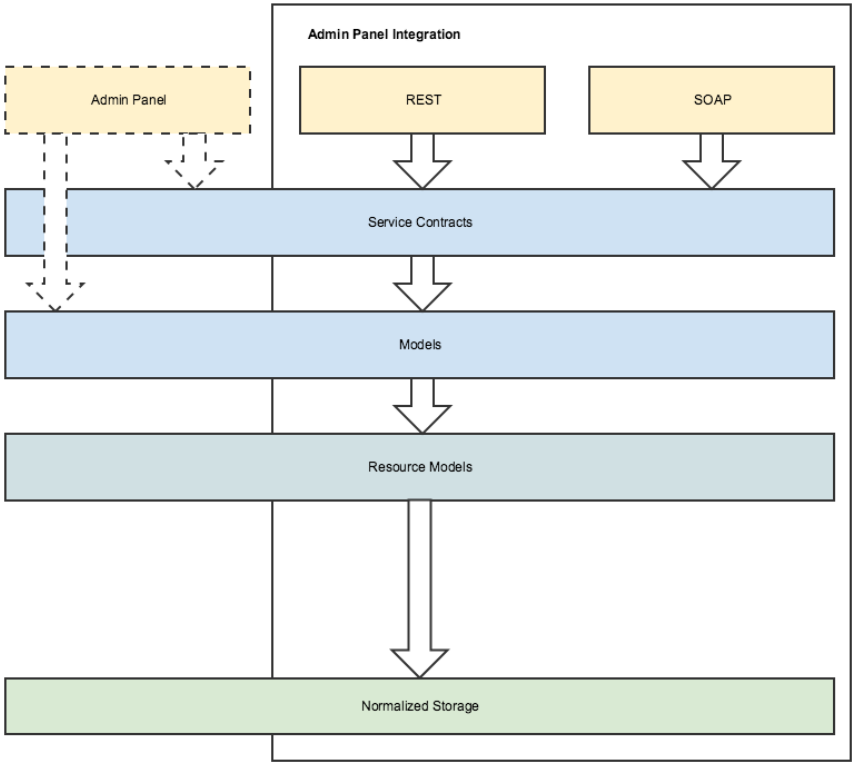
© 2018 Magento, Inc.

SOAP

- ✅ Designed for systems integration
- ✅ Token authentication for customers and admins
- ✅ Cookie authentication
- ✅ Anonymous access
- ✅ Schema WSDL
- ✅ Must return 200 HTTP status code in all cases. In case of exceptional situation, error should be returned in the response

© 2018 Magento, Inc.

Web API – Service Contracts



- Interfaces used for data serialization/deserialization

- Stable interfaces

© 2018 Magento, Inc.

GraphQL Frontend

Why GraphQL?

REST API

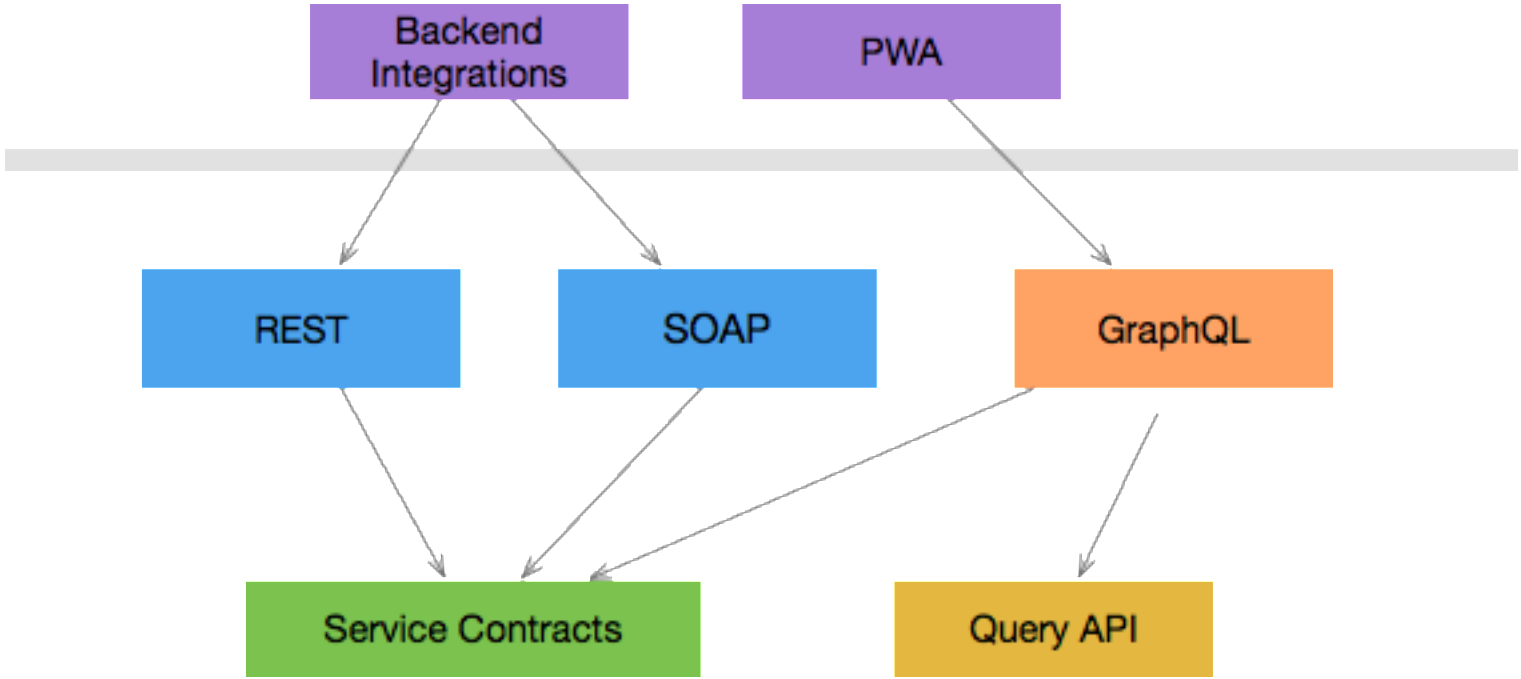
- One request per resource
- You always get all fields
- There is no way to deprecate fields
- There is no way to know what fields clients use
- There is no schema for the data

GraphQL

- Has strictly typed Schema describing all data types and queries
- Documentation is a first class citizen
- Deprecation is a first class citizen
- IDE support
- There are clients for all major languages. Javascript is represented by Apollo, Relay, etc

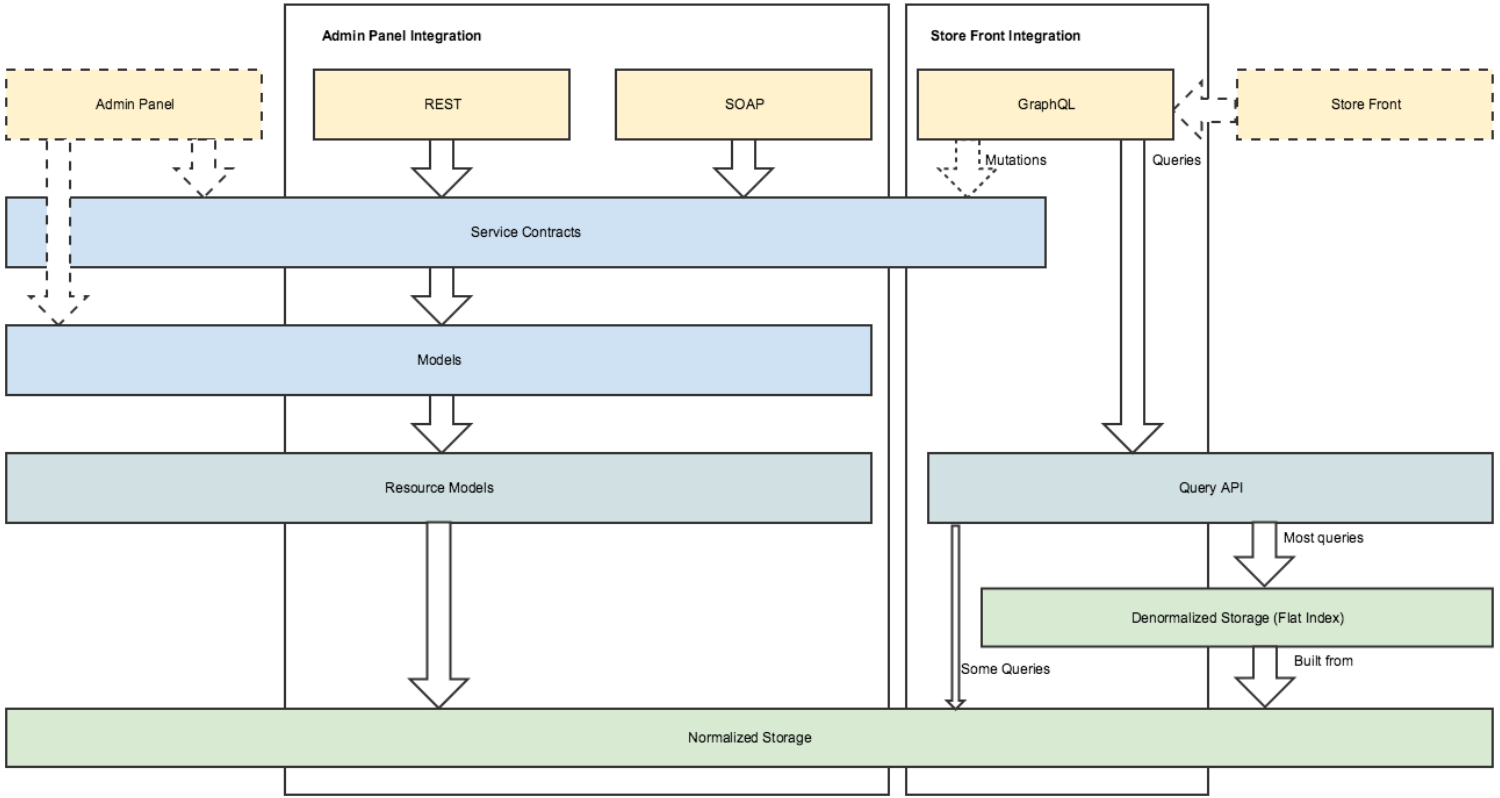
© 2018 Magento, Inc.

GraphQL in Magento, Technical Vision



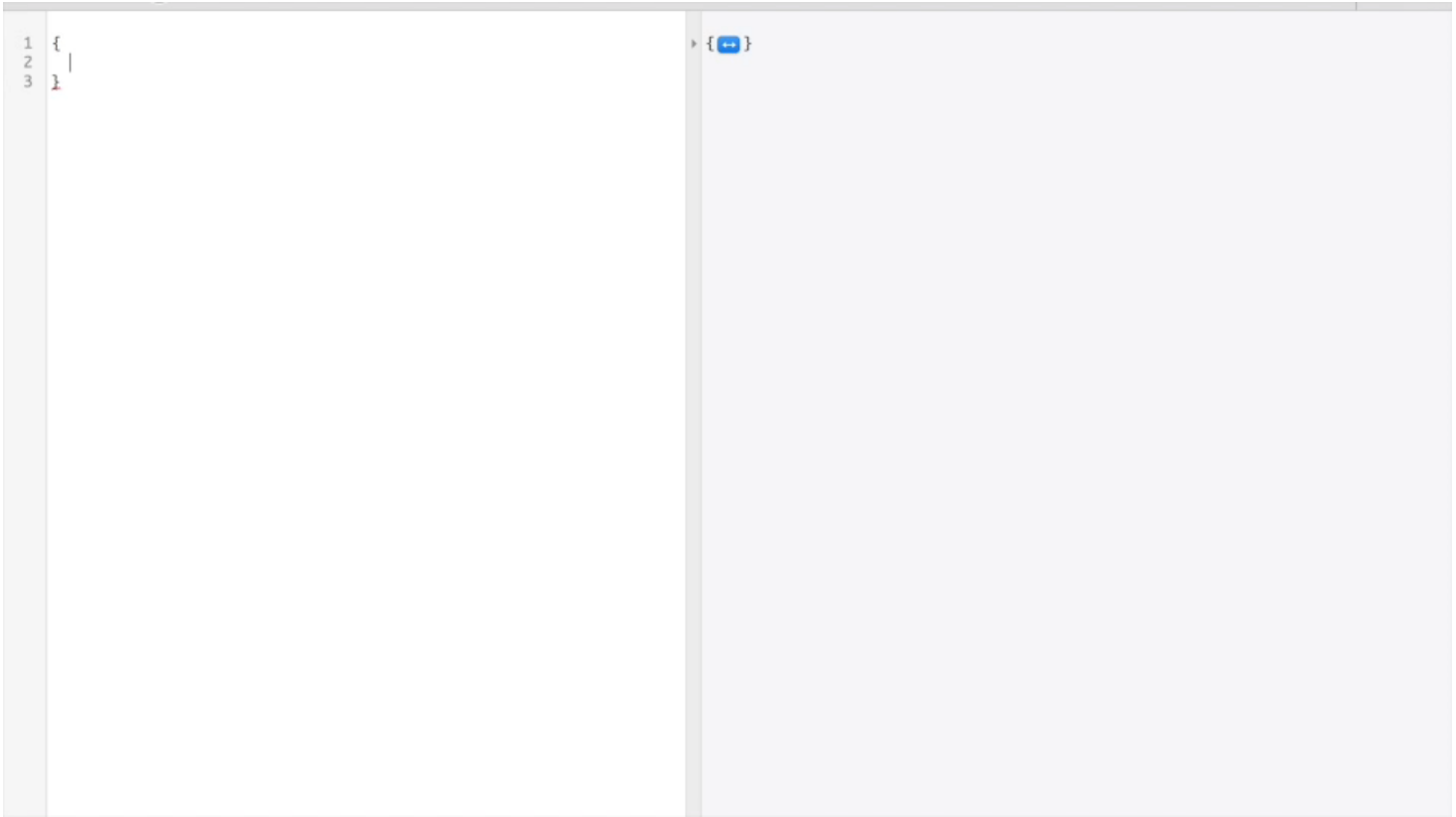
© 2018 Magento, Inc.

Web API – Request Processing – High Level Overview



© 2018 Magento, Inc.

Magento GraphQL - Demo



© 2018 Magento, Inc.

GraphQL. Multiple Queries

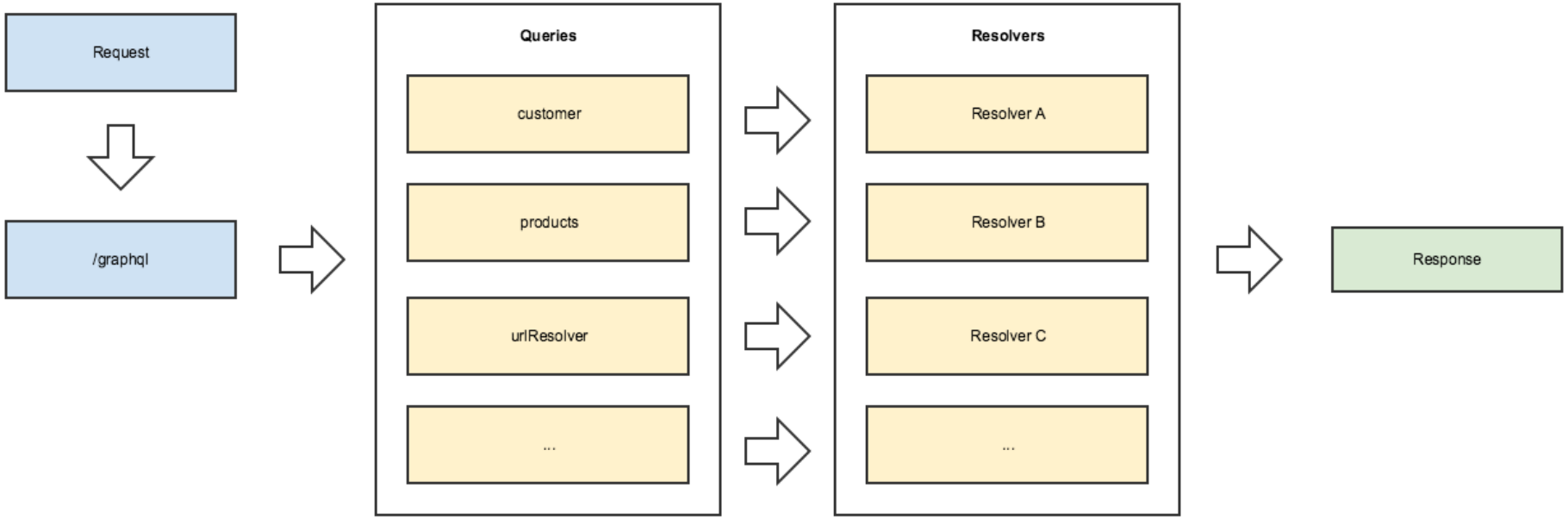
GraphiQL ▶ Prettify History http://localhost.local/project-community-edition/graphql Set endpoint

```
{
  products(
    filter: {
      sku: {eq: "24-MB06"}
    }
  ) {
    items{
      sku
      name
      url_key
      price {
        minimalPrice {
          amount {
            currency
            value
          }
        }
      }
    }
  }

  customer {
    firstname
    lastname
    email
  }

}
```

```
{
  "data": {
    "products": {
      "items": [
        {
          "sku": "24-MB06",
          "name": "Rival Field Messenger",
          "url_key": "rival-field-messenger",
          "price": {
            "minimalPrice": {
              "amount": {
                "currency": "USD",
                "value": 45
              }
            }
          }
        }
      ]
    },
    "customer": {
      "firstname": "Oleksii",
      "lastname": "Korshenko",
      "email": "okorshenko@magento.com"
    }
  }
}
```

© 2018 Magento, Inc.

GraphQL Routing



© 2018 Magento, Inc.

Schema

Root type query: Query

```
customer: Customer

products(
    search: String
    filter: ProductFilterInput
    pageSize: Int
    currentPage: Int
    sort: ProductSortInput
): Products

customAttributeMetadata(attributes: [AttributeInput!]!): CustomAttributeMetadata

urlResolver(url: String!): EntityUrl
```

© 2018 Magento, Inc.

Configuration and Extensibility

File: /vendor/magento/module-customer-graph-ql/etc/graphql.xml

```
<config …>
  <type xsi:type="OutputType" name="Query">
    <field xsi:type="ObjectOutputField"
           name="customer"
           type="Customer"
           resolver="Magento\CustomerGraphQl\Model\Resolver\Customer”/>
  </type>
  <type xsi:type="OutputType" name="Customer">
    <field xsi:type="ScalarOutputField" name="firstname"  type="String"  description="The customer's first name"/>
    <field xsi:type="ScalarOutputField" name="middlename" type="String"  description="The customer's middle name"/>
    <field xsi:type="ScalarOutputField" name="lastname"   type="String"  description="The customer's family name"/>
    <!--....-->
  </type>
</config>
```

Configuration and Extensibility

File: /vendor/magento/module-url-rewrite-graph-ql/etc/graphql.xml

```
<config>
    <type xsi:type="OutputType" name="Query">
        <field xsi:type="ObjectOutputField" name="urlResolver" type="EntityUrl"
               resolver="Magento\UrlRewriteGraphQl\Model\Resolver\UrlRewrite">
            <argument xsi:type="ScalarArgument" name="url" type="String" required="true"/>
        </field>
    </type>
    <type xsi:type="OutputType" name="EntityUrl">
        <field xsi:type="ScalarOutputField" name="id" type="Int" description="..."/>
        <field xsi:type="ScalarOutputField" name="canonical_url" type="String" description="..."/>
        <field xsi:type="ObjectOutputField" name="type" type="UrlRewriteEntityTypeEnum" description="..."/>
    </type>
    <type xsi:type="Enum" name="UrlRewriteEntityTypeEnum"/>
</config>
```

Resolvers

- A resolver performs GraphQL request processing
- It is responsible for:
  - constructing a query
  - fetching data
  - performing any calculations
  - transforming the fetched and calculated data into a GraphQL array format
  - finally, it returns the query results for rendering

© 2018 Magento, Inc.

SQL Mapping: GraphQL Query

```
{
  MagentoCart(
    find:{
      or: {
        items: {sku: {eq: "159d6942b6e92f"}},
        and: {
          customer_id: {gt: 100, lt: 300},
          items: {qty: {eq: "2"}},
        }
      }
    },
    limit: 2,
    sort: {items: {sku:ASC}}
  ) {
    increment_id
    customer_id
    items{
      sku
      qty
    }
  }
}
```

© 2018 Magento, Inc.

SQL Mapping: SQL Query

```
SELECT
    `example_cart_item`.`cart_item_id` AS `c60aa63e3cc2ba068fd5774dc4109d1dca484062`,
    `example_cart_item`.`sku` AS `9b9ae2f63d585113b1724ca0f672261123f30478`,
    `example_cart_item`.`qty` AS `0d7ee4f69deb3a46bd416024a5c4d0d89cde5370`,
    `example_cart`.`cart_id` AS `a7fc996fab2684e93df3866d999a2a9e08edd00c`,
    `example_cart`.`increment_id` AS `6d0cab197ea935b18f62c04fb9acfd6f76758611`,
    `example_cart`.`customer_id` AS `e6b9adbda979922df31f54c8b8c1f2503e217254`,
    `example_cart_item`.`cart_id` AS `5fceb1c437c5ed8a129b77092084f710d7715e4e`
FROM
    `example_cart`
        LEFT JOIN
    `example_cart_item` ON example_cart.cart_id = example_cart_item.cart_id
WHERE
    (example_cart.cart_id IN (SELECT
            `f`.`cart_id`
        FROM
            (SELECT
                `example_cart`.`cart_id`
            FROM
                `example_cart`
            WHERE
                (example_cart.cart_id IN (SELECT
                        `example_cart`.`cart_id`
                    FROM
                        `example_cart`
                    LEFT JOIN `example_cart_item` ON example_cart_item.cart_id = example_cart.cart_id
                    WHERE
                        (((`example_cart_item`.`sku` = '159d6942b6e92f'
                            OR (`example_cart`.`customer_id` > '100'
                            AND `example_cart`.`customer_id` < '300'
                            AND `example_cart_item`.`qty` = '2'))))
                    ORDER BY example_cart_item.qty))
            LIMIT 2) AS `f`))
```

SQL Result

```
+----------+----------------+----------+----------+----------------+----------+----------+
| c60aa... | 9b9ae...       | 0d7ee... | a7fc9... | 6d0ca...       | e6b9a... | 5fceb... |
+----------+----------------+----------+----------+----------------+----------+----------+
|       39 | 159dc90139c5ac | 1        |       39 | 159dc90139c575 |      144 |       39 |
|       40 | 159dc90139c61a | 2        |       39 | 159dc90139c575 |      144 |       39 |
|       88 | 159dc90139dab0 | 3        |       87 | 159dc90139da42 |      104 |       87 |
|       89 | 159dc90139db1e | 2        |       87 | 159dc90139da42 |      104 |       87 |
+----------+----------------+----------+----------+----------------+----------+----------+
```

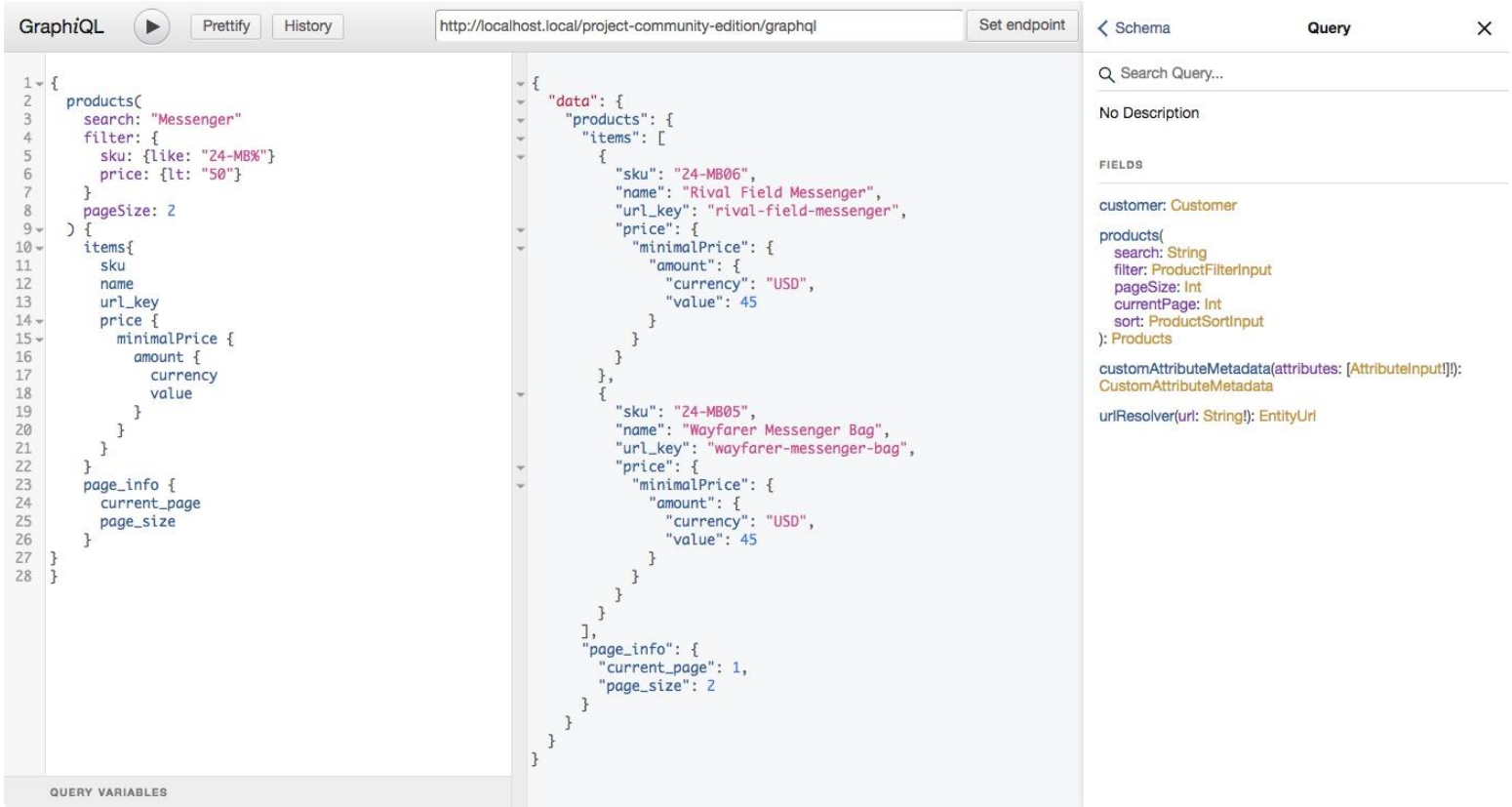
© 2018 Magento, Inc.

GraphQL Response


```
{
  "data": {
    "MagentoCart": [
      {
        "increment_id": "159dc90139c575",
        "customer_id": 144,
        "items": [
          {
            "sku": "159dc90139c5ac",
            "qty": "1"
          },
          {
            "sku": "159dc90139c61a",
            "qty": "2"
          }
        ]
      },
      {
        "increment_id": "159dc90139da42",
        "customer_id": 104,
        "items": [
          {
            "sku": "159dc90139dab0",
            "qty": "3"
          },
          {
            "sku": "159dc90139db1e",
            "qty": "2"
          }
        ]
      }
    ]
  }
}
```

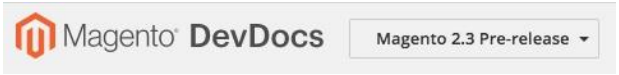

© 2018 Magento, Inc.

Developer Tools. Chrome Extension



© 2018 Magento, Inc.

Documentation



http://devdocs.magento.com/guides/v2.3/graphql/index.html

Thank You!

Oleksii Korshenko
Magento 2 Architect,
Community Engineering
@okorshenko

© 2018 Magento, Inc.