MEDIAFOREUROPE

GraphQLConf 2025

hosted by GraphQL Foundation

One Schema To Rule Them All: Simplifying 10+ Mediaset Apps With One Single GraphQL Service

Marco Reni
Architect

Hello! I’m @marcoreni



- Architect @ Mediaset (MFE)
- Oversee FE and BfF architectures and developments
- Backend / DevOps background
- ❤️ to take selfies with animals

MEDIAFOREUROPE

100% MEDIASET

100% MEDIASETespaña.

75% ProSiebenSat.1 Media SE

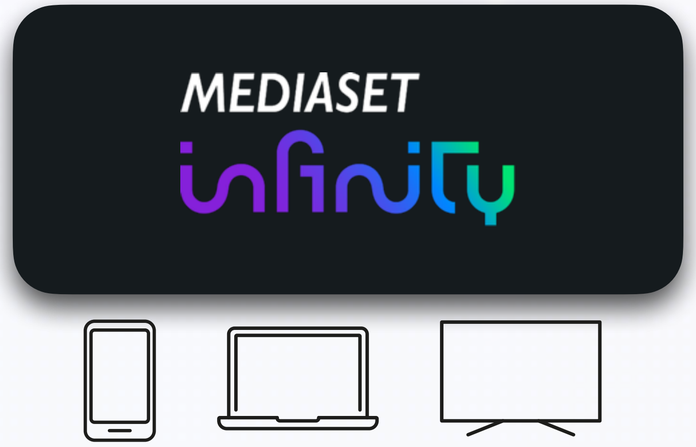
MFE
MEDIAFOREUROPE



Agenda

- Scope of the project
- Designing the schema
- Performance & Scalability strategies
- Now & Next

Properties in scope

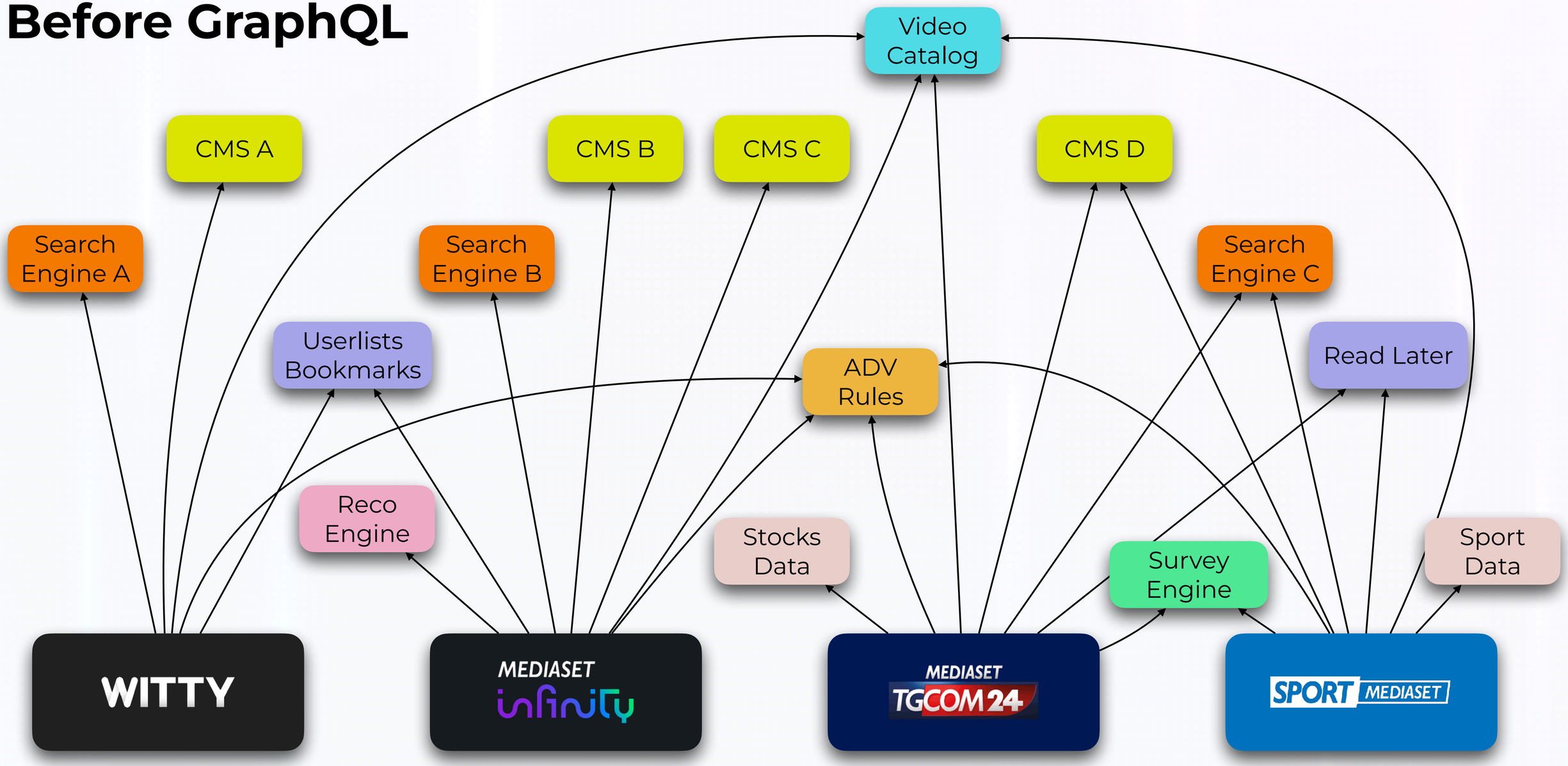




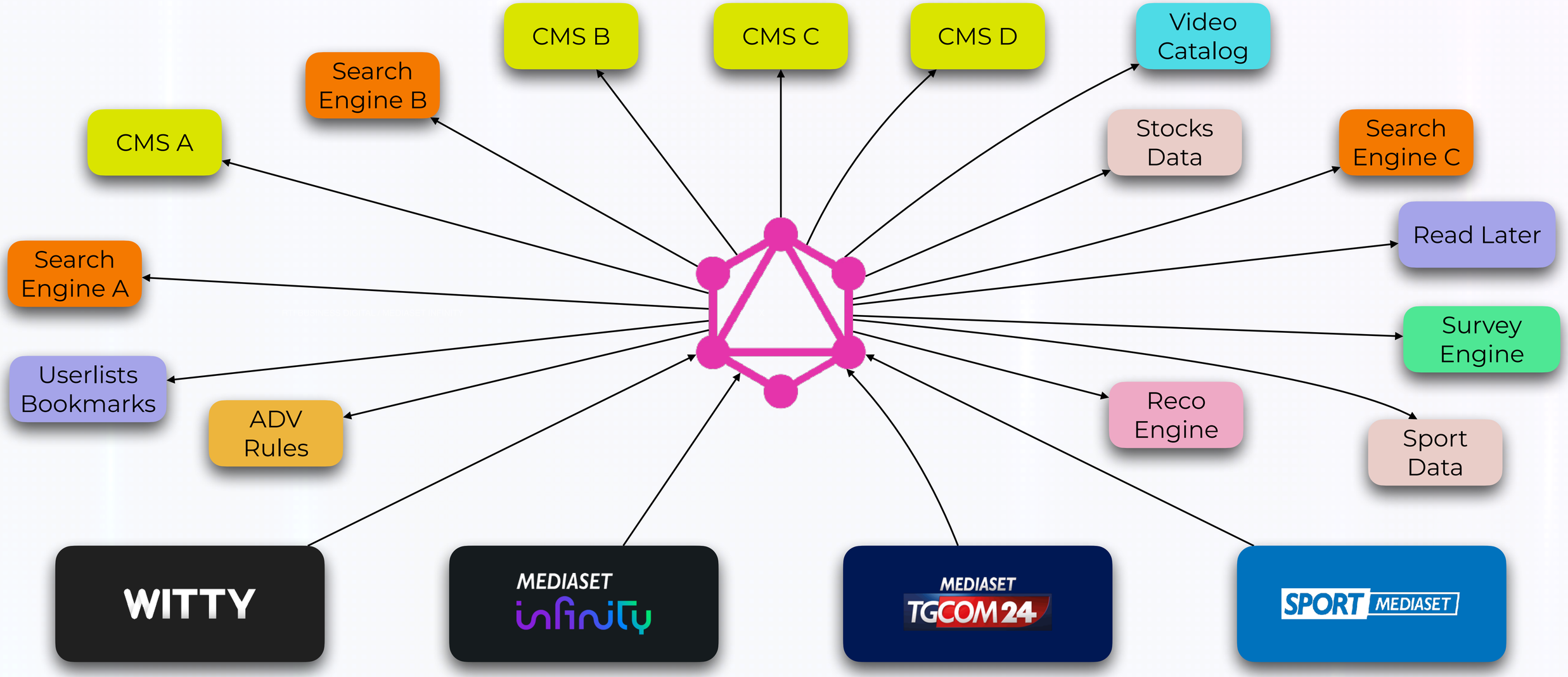




Before GraphQL



Target Architecture



Requirements

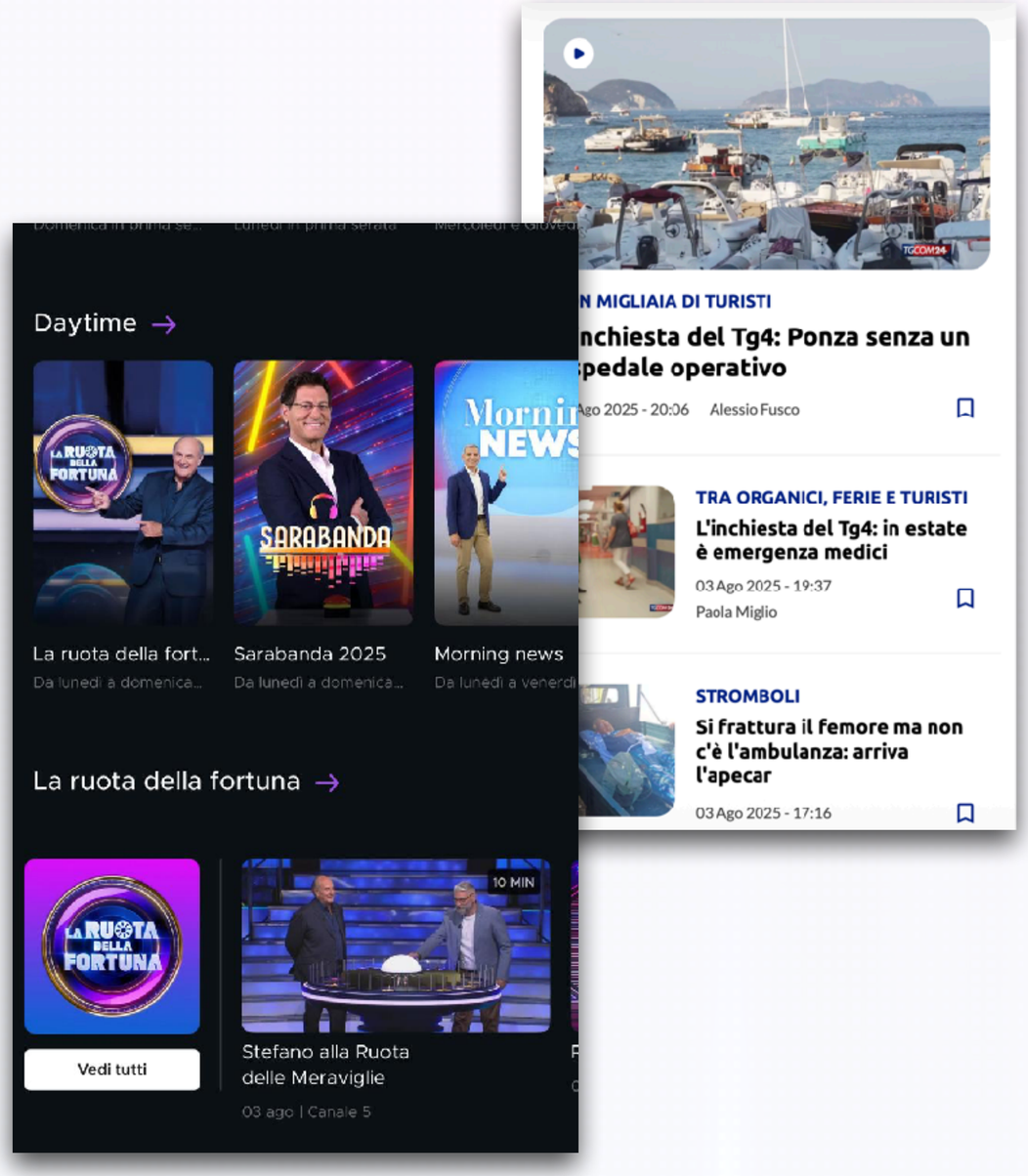
- Unique schema for all properties
- Handle backends failures
- Resilient to spikes in traffic
- Capability to integrate all backends in all properties

Designing the schema

Building the schema

Principles

- Start from the atoms
- Schema types describe the content, not the representation of it
- Templating attributes
- Don’t know what to do with it? Ignore it!



Building the schema

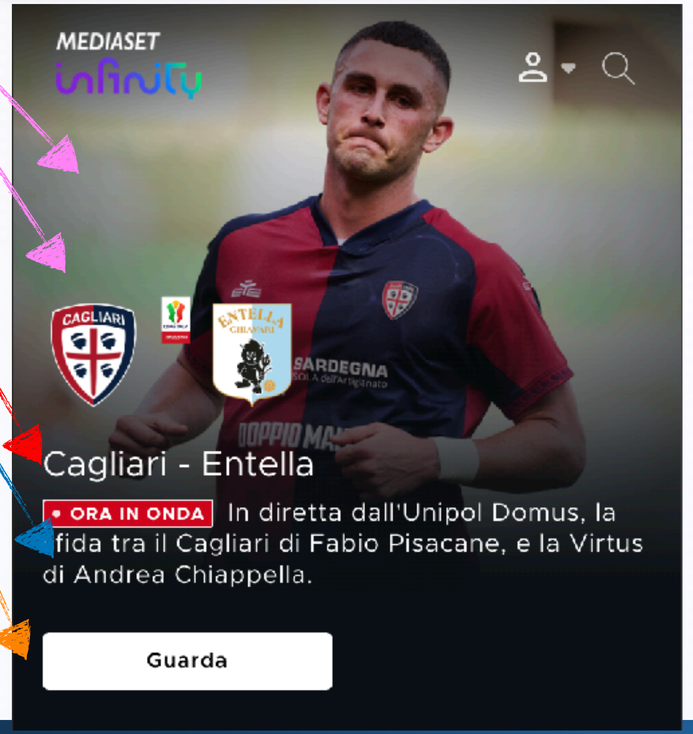
Items

```
interface Item {
    id: ID
    cardEyelet: String
    cardTitle: String
    cardText: String
    cardImages: [ImageUnion!]
    cardLink: Link
    cardAttributes: CardAttributes
    cardCtas: [VisualLink!]
}

type CardAttributes {
    flags: [CardFlag!]
    variant: CardVariant
    template: CardTemplate
    bgColor: HexColorCode
    borderColor: HexColorCode
    textColor: HexColorCode
}
```







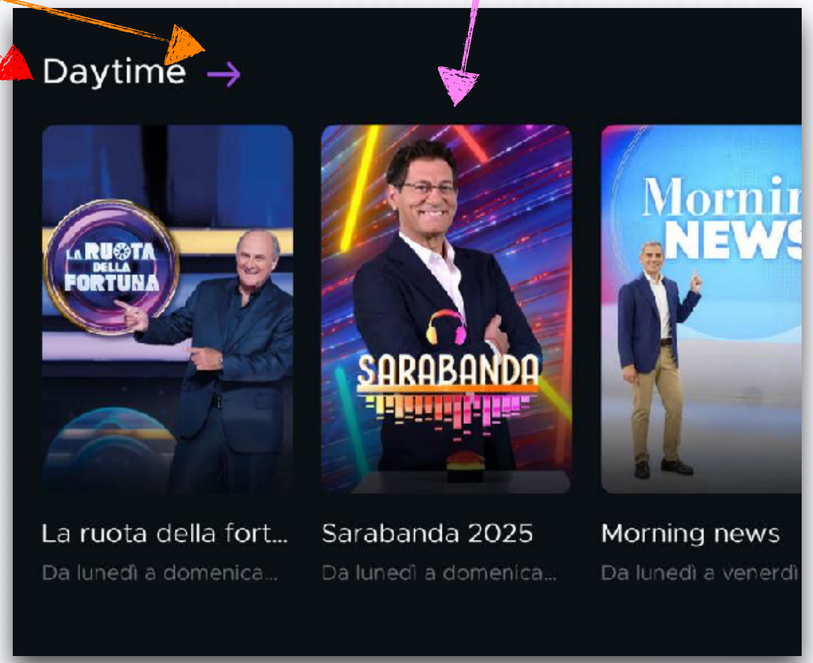
Building the schema

Collections

```
interface Collection {
    id: ID
    title: String
    itemsConnection(first: Int, after: String): ItemsConnection
    link: Link
    ctas: [VisualLink!]
    attributes: CollectionAttributes
    images: [ImageUnion!]
}

type ItemsConnection {
    items: [Item!]
    pageInfo: PageInfo!
}

type PageInfo {
    hasNextPage: Boolean!
    endCursor: String
    context: String
}
```





Building the schema

Sections

```
type Section {
    id: ID
    title: String
    attributes: SectionAttributes
    link: Link
    ctas: [VisualLink!]
    collections: [Collection!]
    images: [ImageUnion!]
}
```



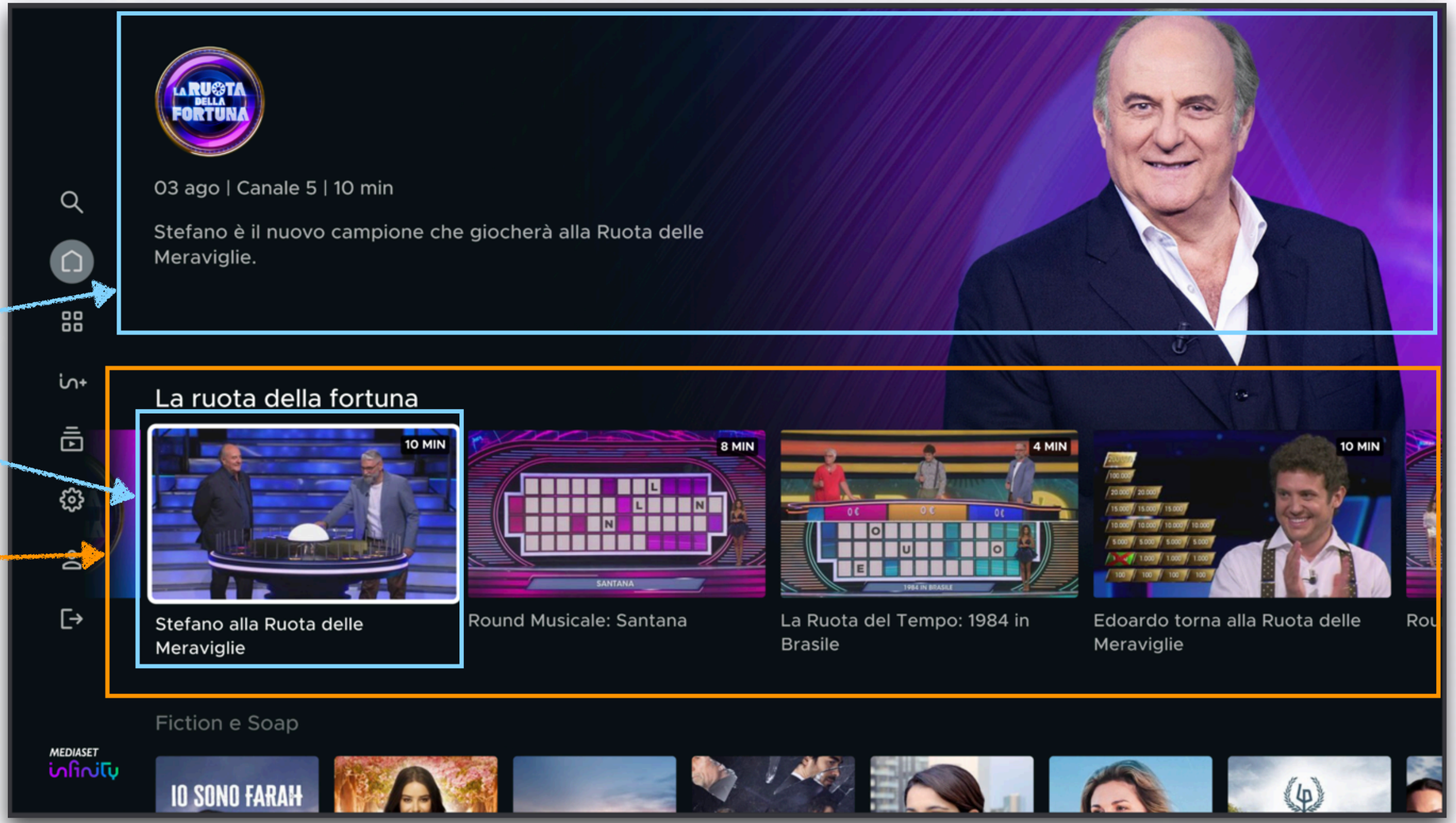


Building the schema

Does this work on TV?

Item

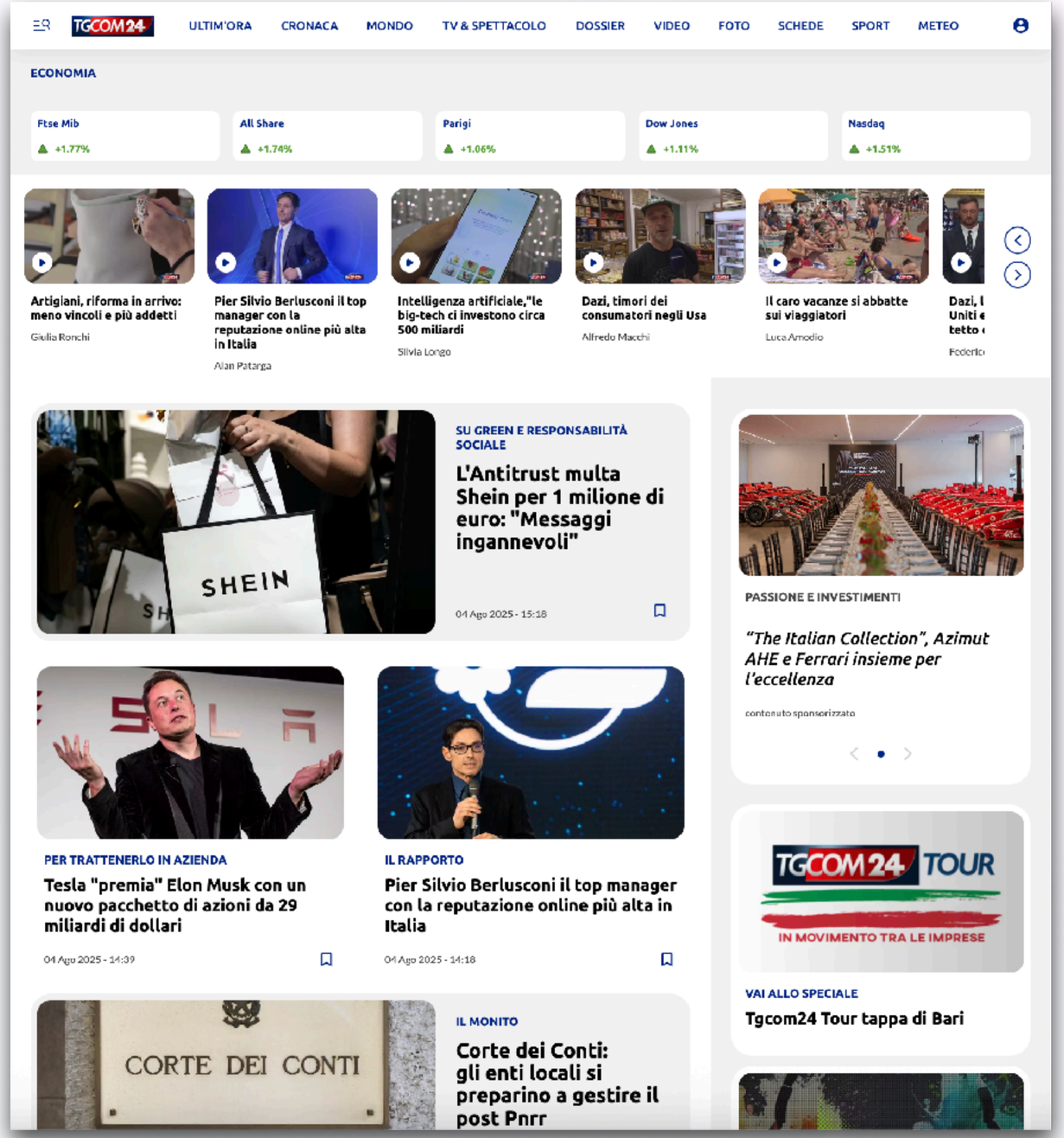
Collection



Building the schema

Tgcom24 Website

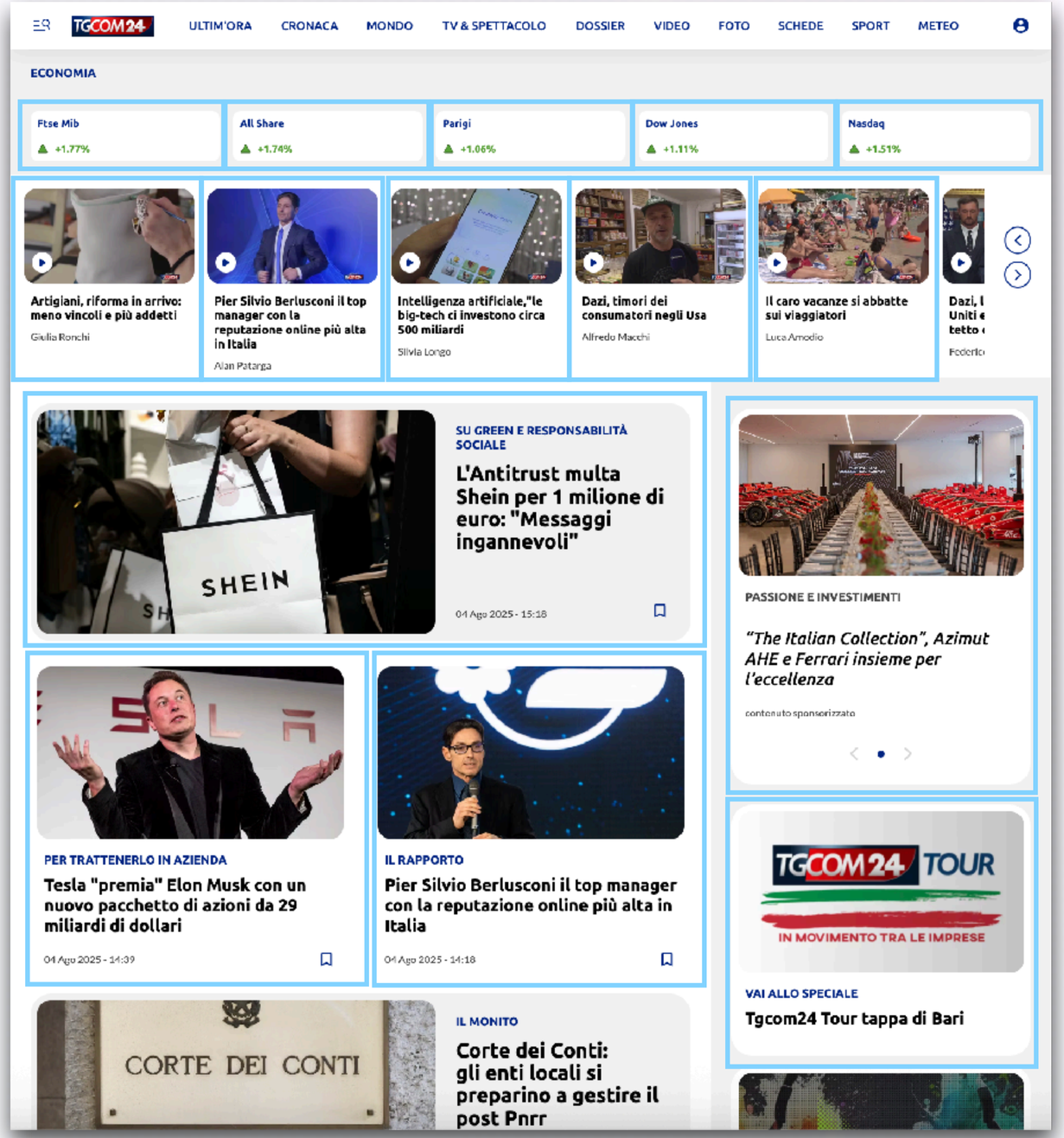
```
query getPage() {
  areaContainers {
    areas {
      sections {
        collections {
          items {
          }
        }
      }
    }
  }
}
```



Building the schema

Tgcom24 Website

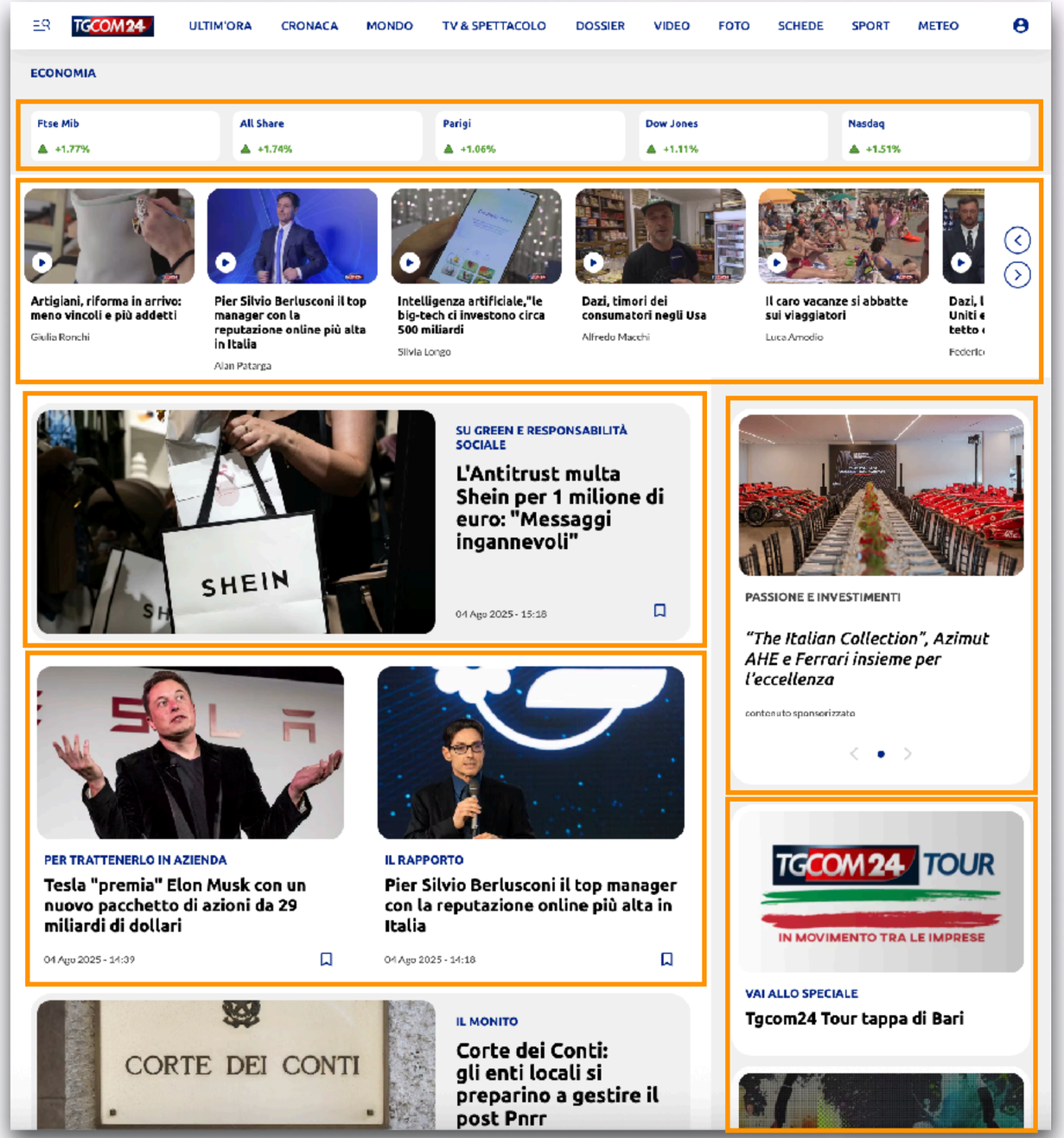
```
query getPage() {
  areaContainers {
    areas {
      sections {
        collections {
          items {
          }
        }
      }
    }
  }
}
```



Building the schema

Tgcom24 Website

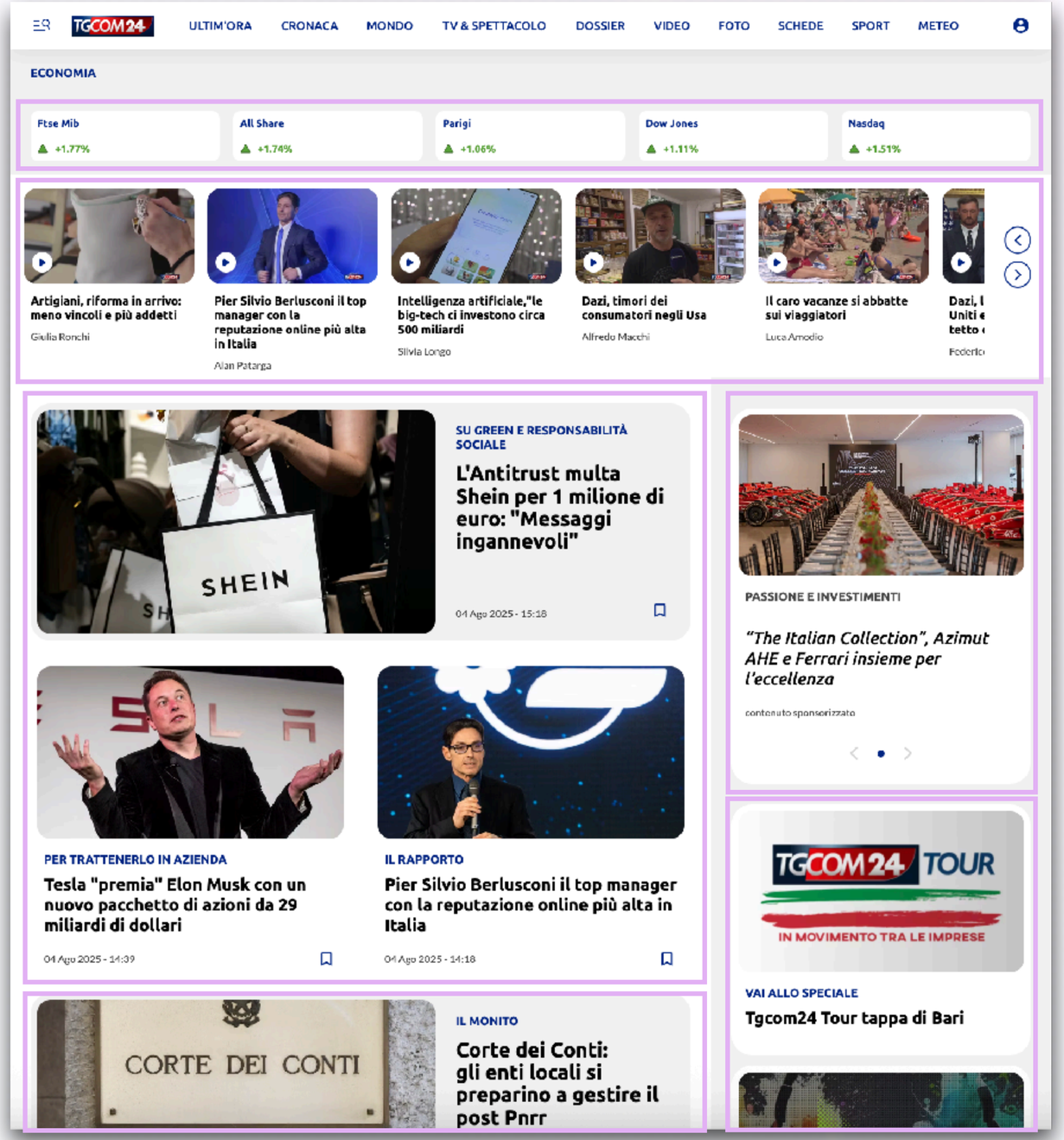
```
query getPage() {
  areaContainers {
    areas {
      sections {
        collections {
          items {
          }
        }
      }
    }
  }
}
```



Building the schema

Tgcom24 Website

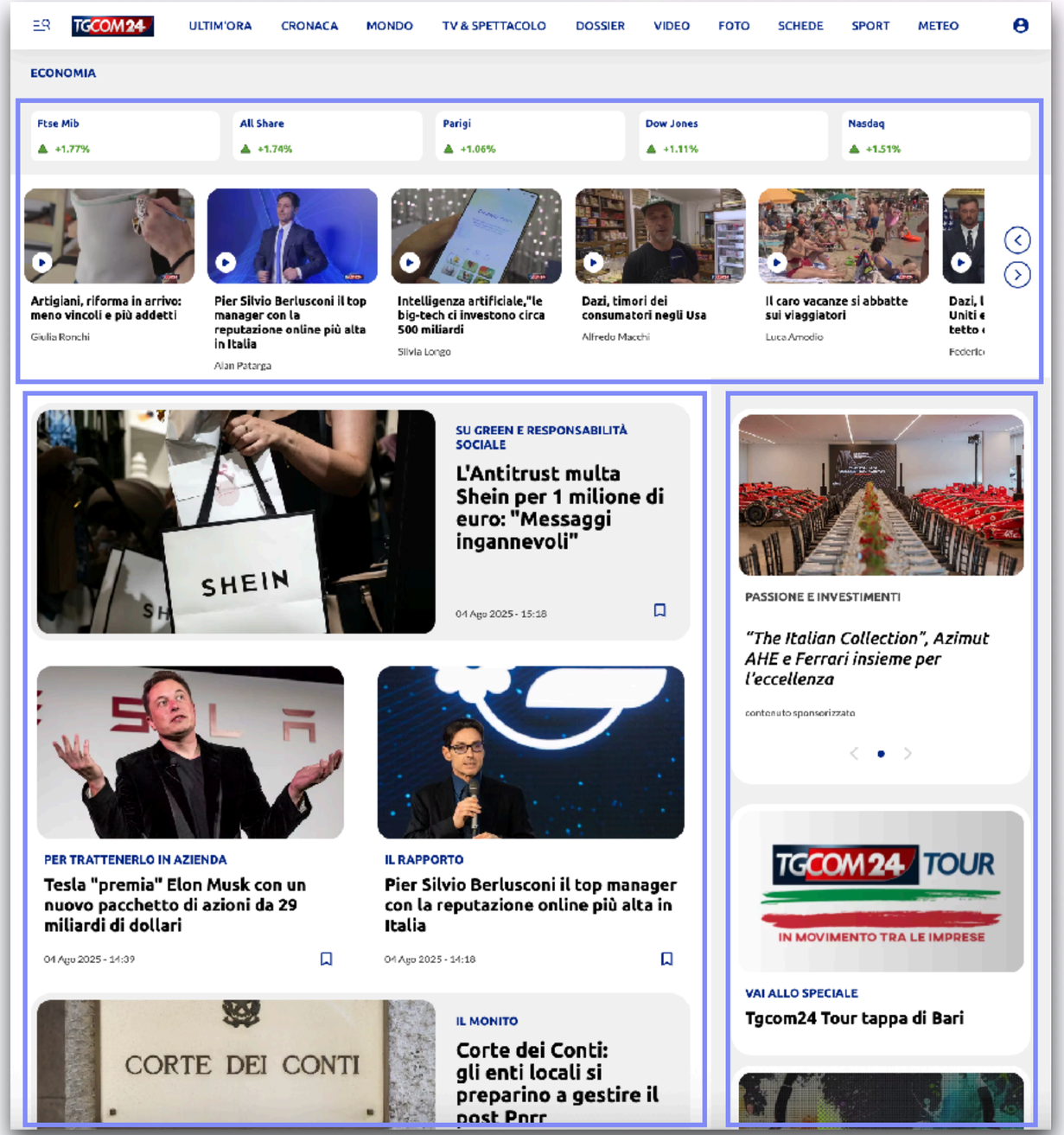
```
query getPage() {
  areaContainers {
    areas {
      sections {
        collections {
          items {
          }
        }
      }
    }
  }
}
```



Building the schema

Tgcom24 Website

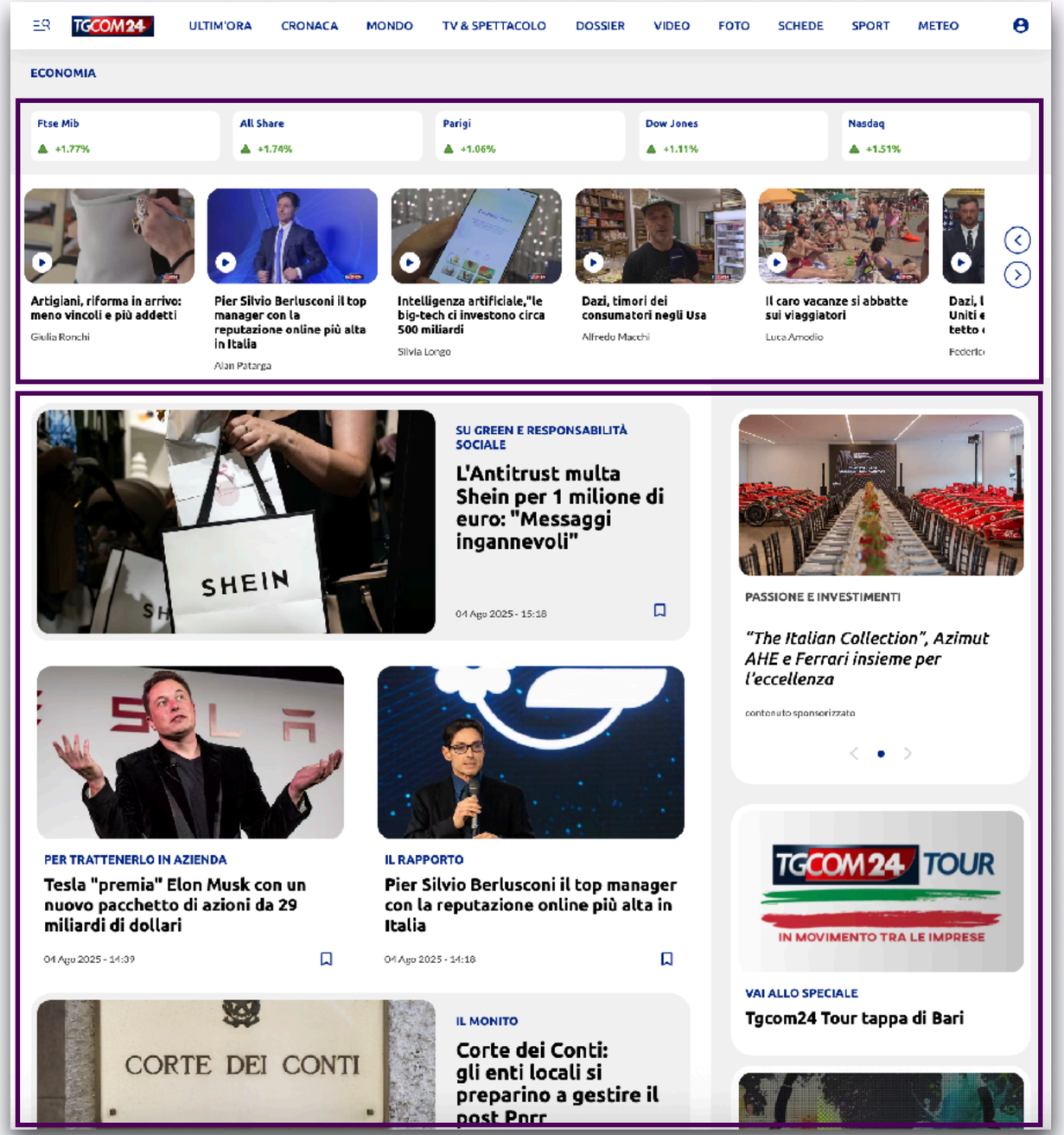
```
query getPage() {
  areaContainers {
    areas {
      sections {
        collections {
          items {
          }
        }
      }
    }
  }
}
```



Building the schema

Tgcom24 Website

```
query getPage() {
  areaContainers {
    areas {
      sections {
        collections {
          items {
          }
        }
      }
    }
  }
}
```



Detail Pages

Avoid repeating stuff

```
query getDetailPage() {
  dataSource {
    ... ItemFragmentForDetailPage
  }
  ...
  items {
    ... on PlaceholderItem {
      resolverType
      resolverParams {
        key
        value
      }
    }
  }
}
```

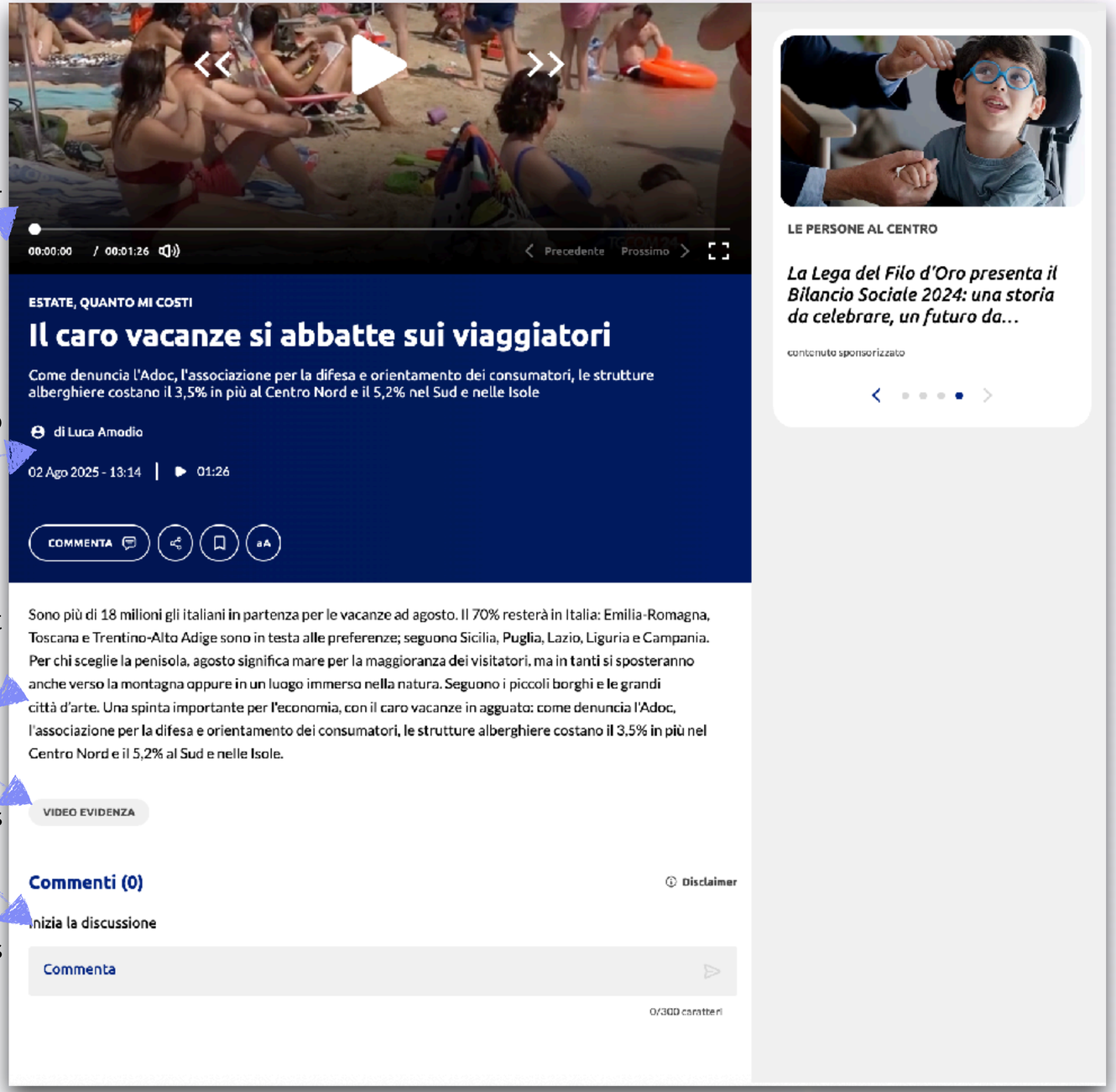
Player

VideoInfo

Content

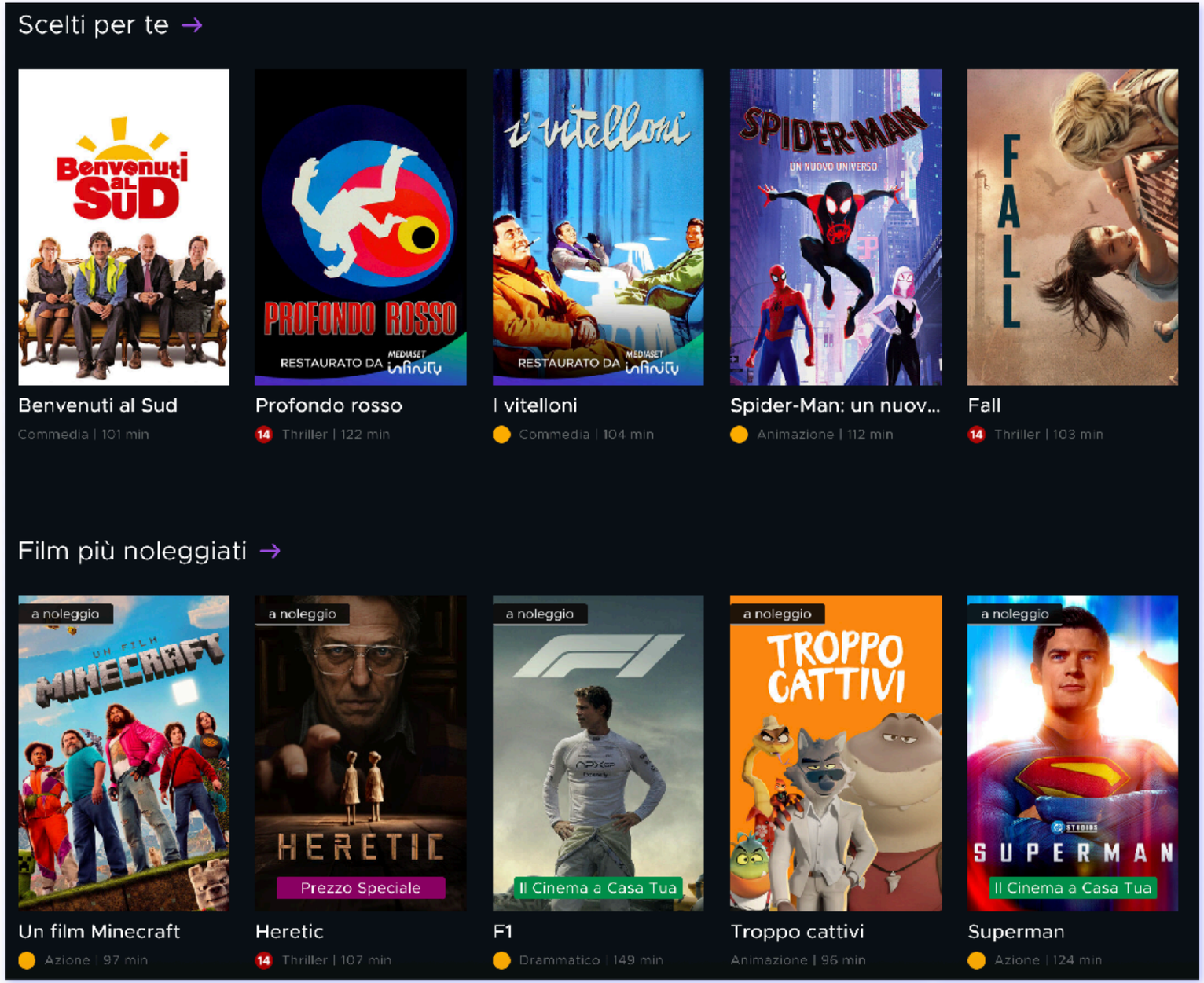
Tags

Comments



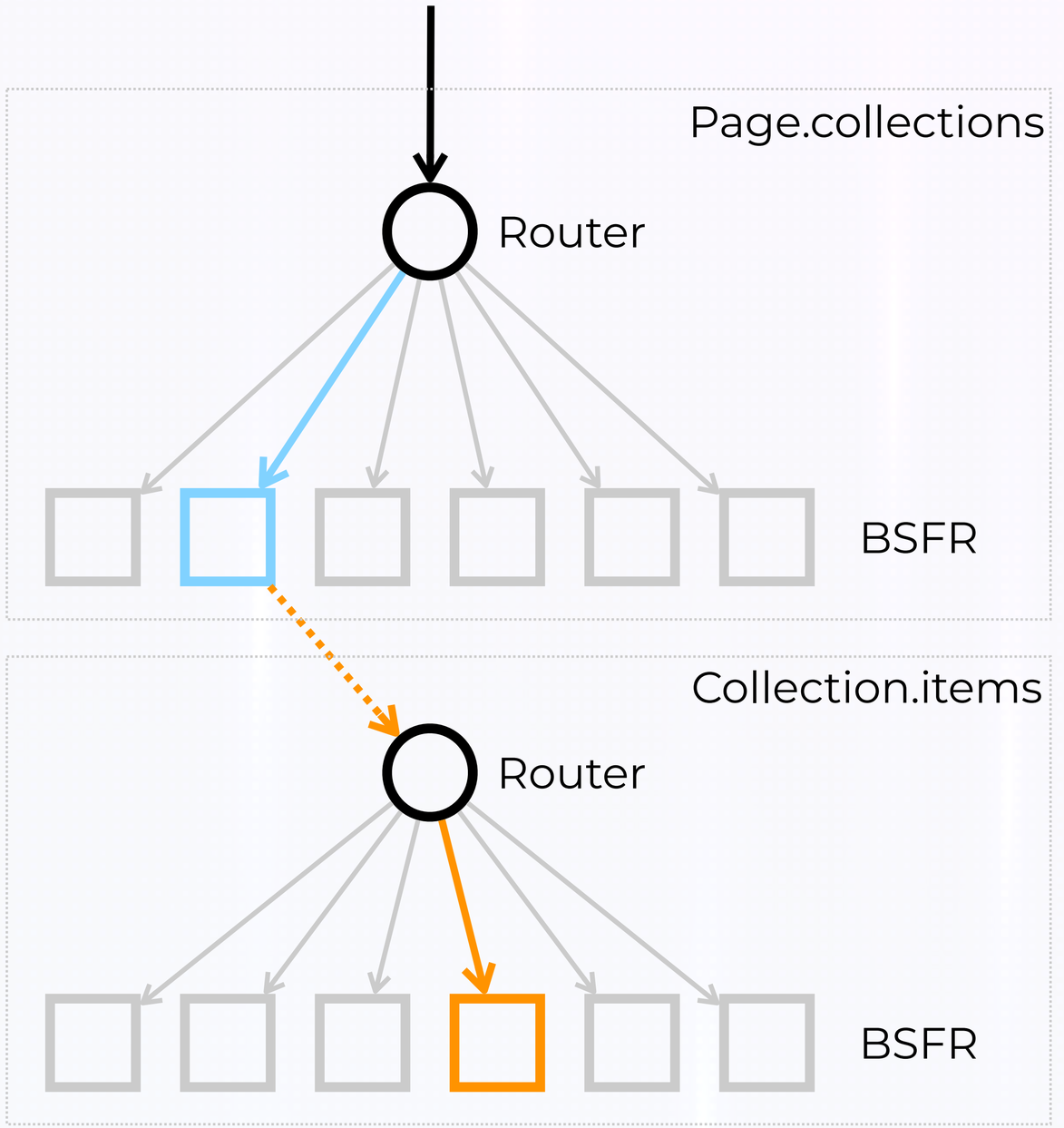
MovieCollection
from Recommendation Engine

MovieCollection
from Video Catalog



Backend-Scoped Field Resolvers

- Each field resolver is a router to a *Backend-Scoped Field Resolver (BSFR)*
- Source carries the resolution *scope*
- We can switch between *scopes* at any time during the resolution tree



Performance & Scalability strategies

Backends Caching

- Two tiered normalized backend cache (In Memory + Valkey)
- Stale-while-revalidate (+ stale-if-error) mechanisms
  - Avoids *Thundering Herd* issues
  - Survives failing backends

Trusted Documents (with a sparkle of DevX)

- “Automatic Persisted Queries” on non-production environments
- “Trusted Documents” on production environments
- Common key-value store of hashed queries filled by non-production environments
- Read-only and GET-only in production

Persisted Documents RFC: https://github.com/graphql/graphql-over-http/pull/264



CDN Cache…. What about Personalized Content?

- Extend Placeholder mechanism to Sections and Collections
- Clients use skeletons while making additional GraphQL requests
- Separate request is not cached but has a much smaller payload





Now & Next

Current Status

- 5 years since first production deployment
- 3500 LoC schema - 144 types - 48 queries
- 5 production deployments with same codebase
- 35M avg requests “post-CDN” served per day across all deployments

Pros...

- Greatly improved page-load and user experience
- Integrated new functionalities on clients with almost zero changes
- BONUS! Switched CMS with zero changes on clients

... and Cons

- Trusted Documents are lazily loaded on first request, for each container
- Lots of nesting in the schema, leads to processing bottlenecks on lower-end devices
- InMemory backend cache is not synchronized between containers

Future Developments

- Pre-loading and pre-computing Most Frequently Used Trusted Documents
- Simplify page structure for simpler apps (eg. TV)
- Implement InMemory cache synchronization between containers

Thanks!

marco.reni@mediaset.it
@marcoreni