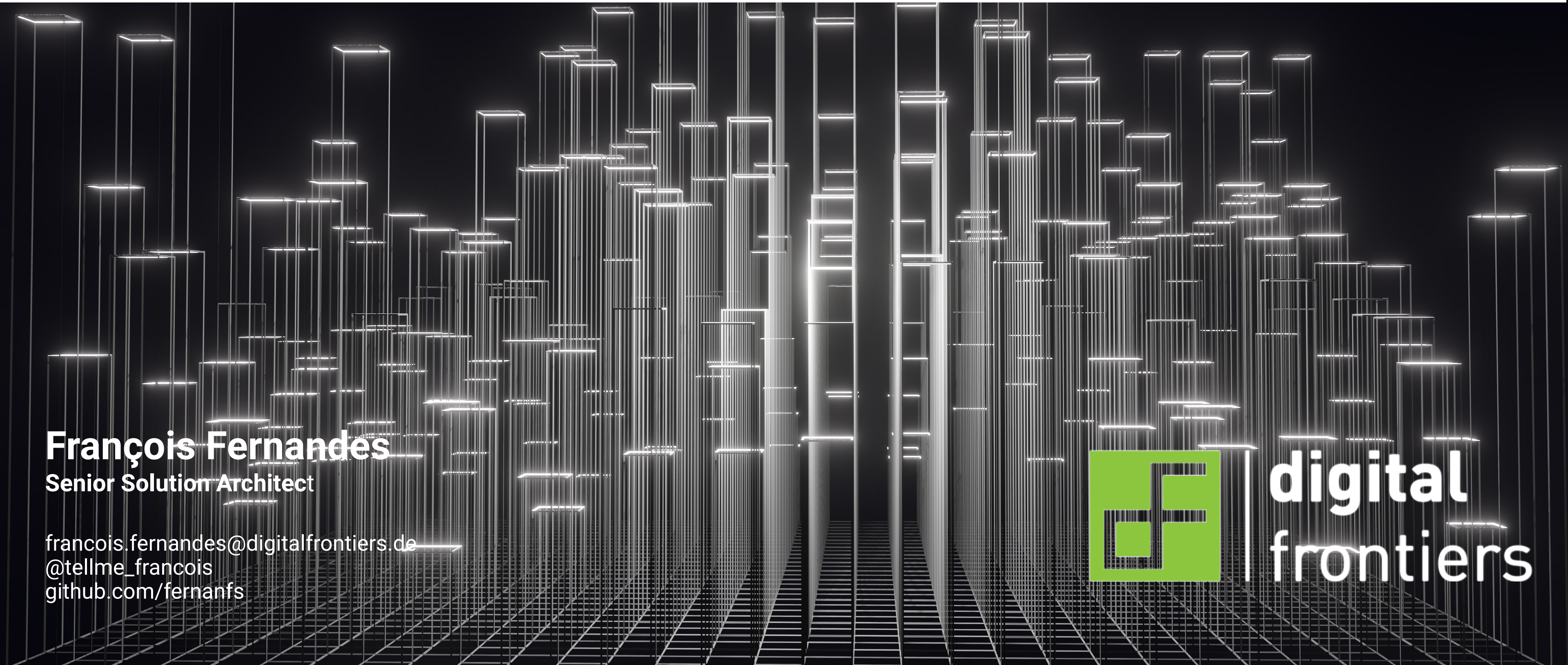
PDF with CQRS/ES

Combining PDF with modern Software Architecture



François Fernandes
Senior Solution Architect

francois.fernandes@digitalfrontiers.de
@tellme_francois
github.com/fernanfs

digital frontiers

What is CQRS/ES?

CQRS ES

Command &

Q
R
S

E
S

digital frontiers

Command & Query R S E S

Command &
Query
Responsibility
S
E
S

Command &
Query
Responsibility
Segregation

E
S

Command &
Query
Responsibility
Segregation

Event
S

Command &
Query
Responsibility
Segregation

Event
Sourcing

Command & Query Responsibility Segregation

Event Sourcing

What about **PDF?**

Command &
Query
Responsibility
Segregation

Event
Sourcing

Create here

PDF

Command & Query Responsibility Segregation

Event Sourcing

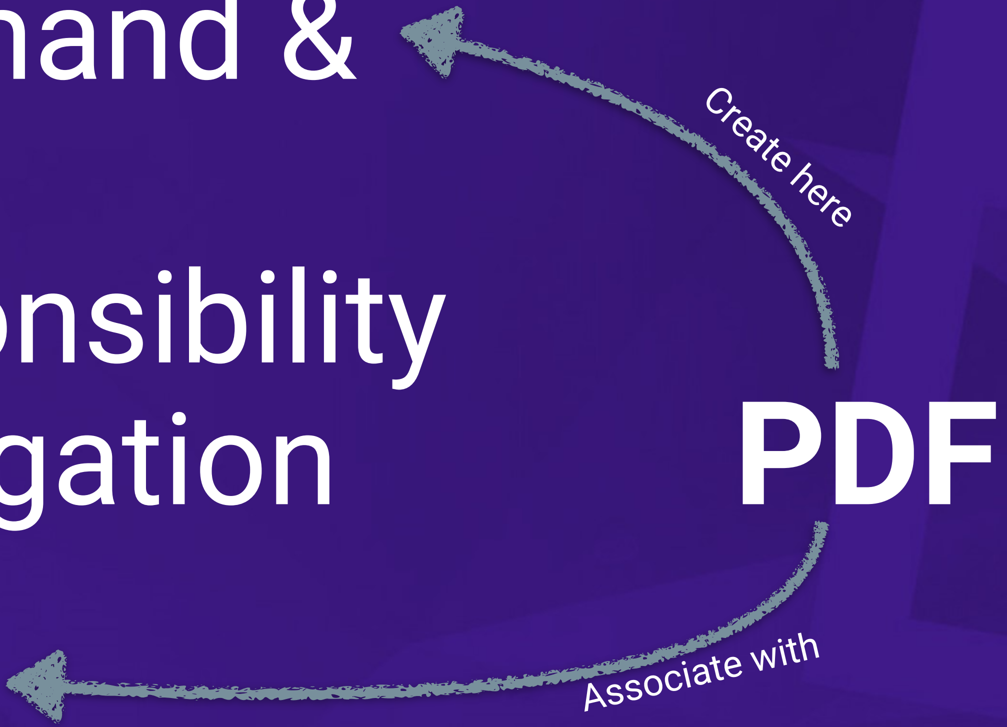


Photo by Simone Secci on Unsplash

Thank you for listening.

Questions?

Photo by Simone Secci on Unsplash

Still unclear?

Well, then…

Challenges in Software Projects

- Infrastructure Landscape is changing rapidly
  - Mixed environments of On-Premise, Private-, Hybrid- or Public-Cloud
  - New concepts like FaaS (Function as a Service)
- Ever increasing loads on applications, even on purely internal applications
- Ever changing requirements, by multiple parties
- Increasing governance requirements in many industries
- Analytics and Machine Learning require a lot of additional data

CQRS/ES

CQRS/ES

Event Sourcing

Event Sourcing

- In classical applications, the source of truth is typically a SQL database, with individual tables representing the current state
    - History of changes often generated by the application or DB-Triggers
- With Event Sourcing, every change is represented as an event
    - The current truth is, the summary of all events

Lets' look at an example!

Sequence of Events

- CreateContactEvent

- A new business contact is created
- In event sourcing there is typically one event that represents the creation of an entity

CreateContactEvent

```
{
  "FirstName": "Maria",
  "LastName": "Strong",
  "DateOfBirth": "1979-05-29",
  "Address": {
    "Line1": "Ocean Drive 22",
    "PostalCode": "42420",
    "City": "Nevermind",
    "Country": "NL"
  }
}
```

Sequence of Events

- CreateContactEvent
- NameChangedEvent

- After some time, we are informed that the person got married and changed the last name
- Instead of just changing the data, an event expressing the intent is stored

NameChangedEvent

```
{
  "LastName": "Fisher",
  "AssociatedDocuments": [
    { "ID": "4711", "Name": "IncomingLetter03031.pdf" }
  ]
}
```

Sequence of Events

- CreateContactEvent
- NameChangedEvent
- ContactMovedEvent

- Time is moving again and there seems to be a serious relocation
- Another event is generated, updating the state. Without deleting any existing data and only the required information.

ContactMovedEvent

```
{
  "Address": {
    "Line1": "Salt Plateau 82",
    "PostalCode": "00001",
    "City": "Dontcare",
    "Country": "Mars"
  },
  "AssociatedDocuments": [
    { "ID": "7721", "Name": "IncomingLetter05732.pdf" }
  ]
}
```

Sequence of Events

- CreateContactEvent
- NameChangedEvent
- ContactMovedEvent

Event Stores typically store the event with some metadata, like:
- timestamp of the event
- user initiating the event

… and more

With these events at hand, we can not only see the current state (our truth as of date), but the whole history.

This allows to find answers to questions, that have not been thought of at the time of writing the application:

- Did that person change their name?
  - How often?
- Did that person move?
  - How often?
  - At which interval?
- What was the address of the person at the date X?
- Which documents caused the changes to the contact?

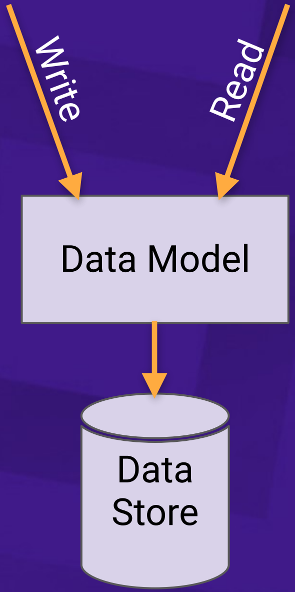
digital frontiers

CQRS/ES

Command & **Q**uery **R**esponsibility **S**egregation

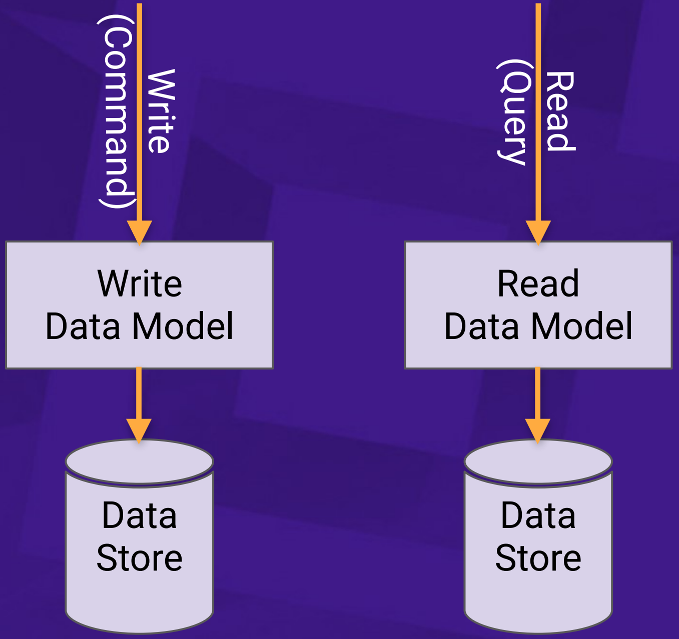
Classical 3-tier architectures

- Typical 3-tier architecture
  - Access to the application data is handled by a data model
  - This data model is shared for read and write access
- That single data model must support all intended usecases



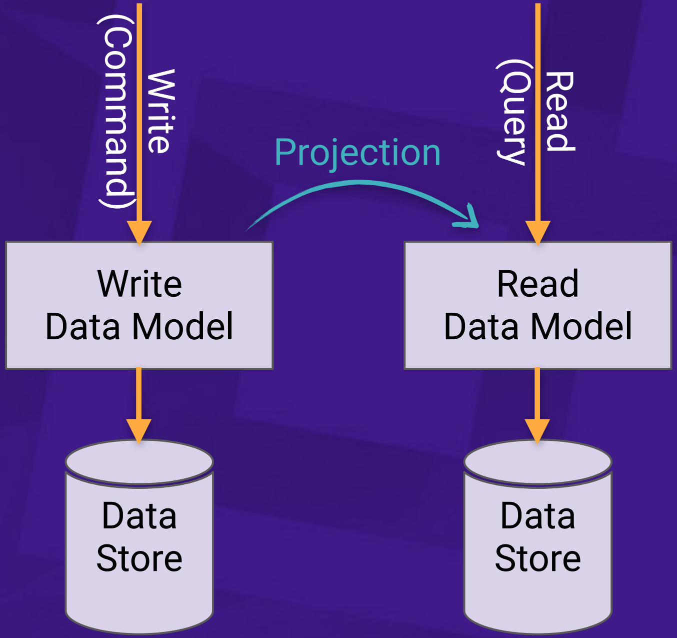
Command & Query Responsibility Segregation

- With CQRS, writes (commands) and reads (queries) are separated
- Commands state what shall be done to the current state
  - The command logic contains the actual business logic
- After a command has been successfully executed, the read model will be derived (Projection)



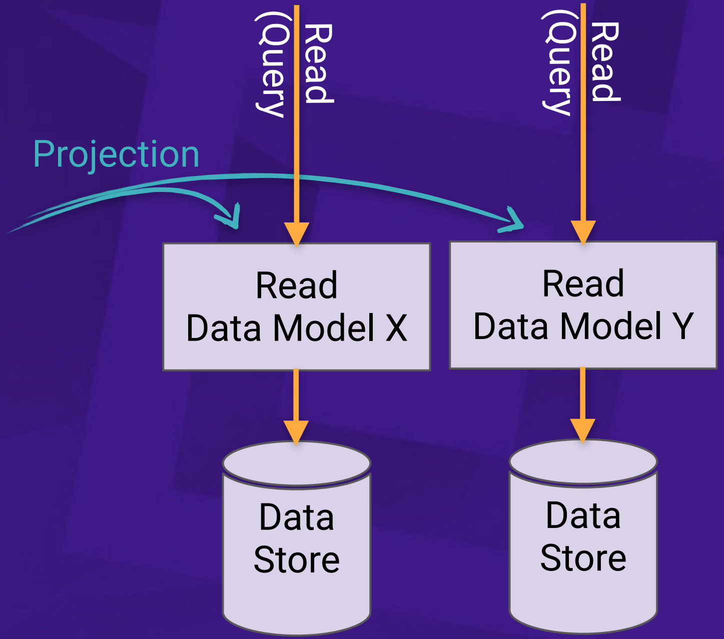
Command & Query Responsibility Segregation

- With CQRS, writes (commands) and reads (queries) are separated
- Commands state what shall be done to the current state
    - The command logic contains the actual business logic
- After a command has been successfully executed, the read model will be derived (Projection)



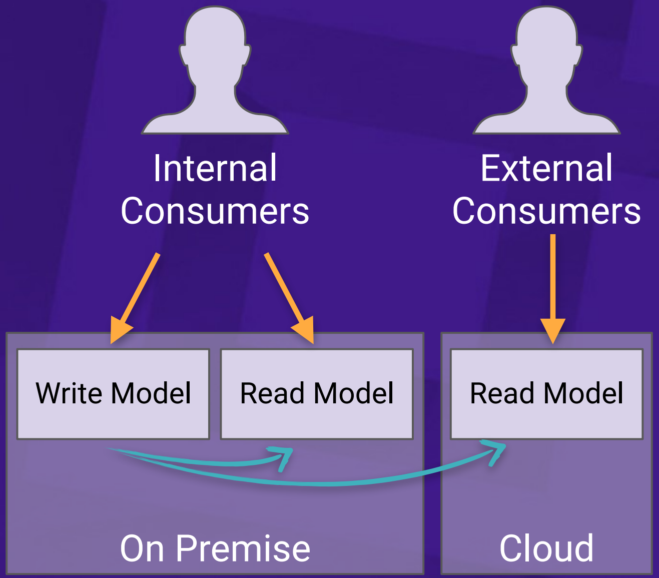
Read Model Projections

- There is no 1-to-1 relationship between write and read models
- There may be as many read models as required
- Read models are created for a specific purpose, best serving the actual use case.



New Architecture Possibilities

- Projections may result in different locations
- Protecting data by exposing what is actually needed
- Every projection is made specifically for the target audience.



CQRS/ES

combining

Command & **Q**uery **R**esponsibility **S**egregation

with

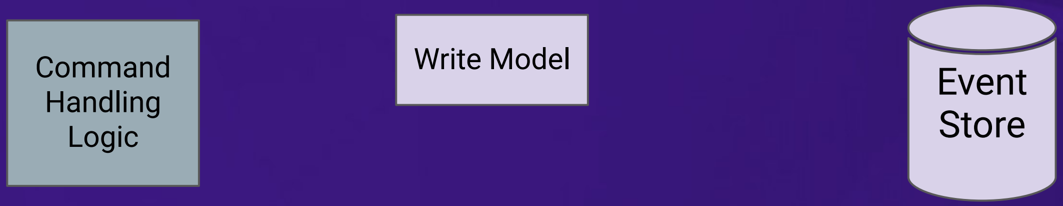
Event **S**ourcing

Combining CQRS with ES

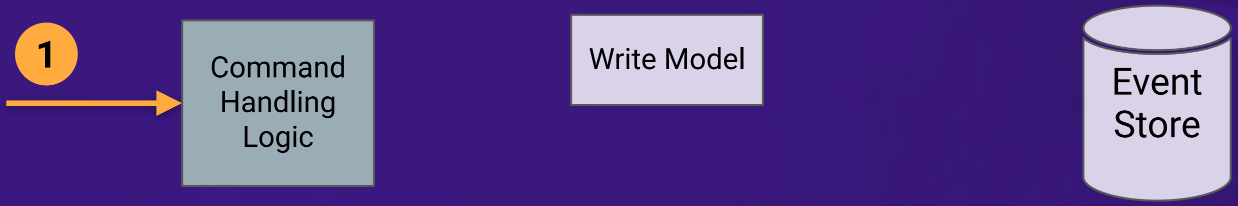
- CQRS doesn’t require Event Sourcing
  … but they complement each other very well!
- Combining both streamlines application design, implementation and maintenance

Let’s see, how

Combining CQRS with ES

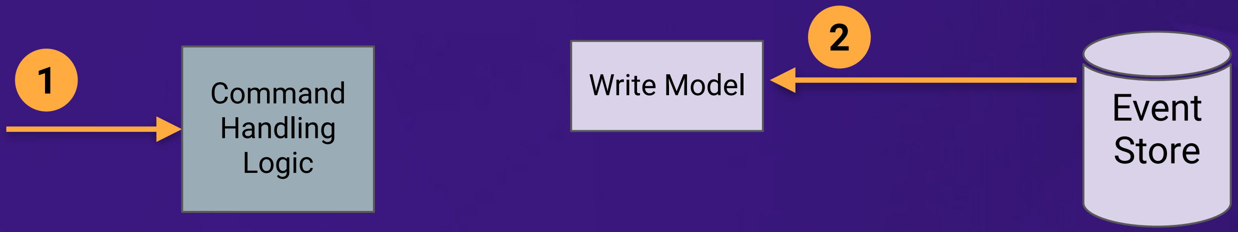


Combining CQRS with ES



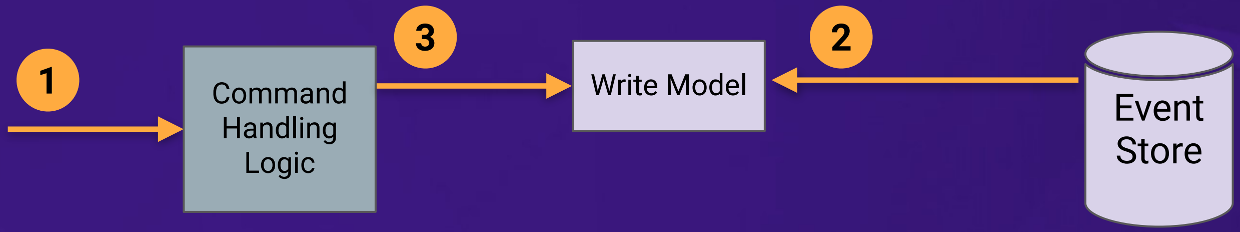
1. A write operation is initiated (Command)

Combining CQRS with ES



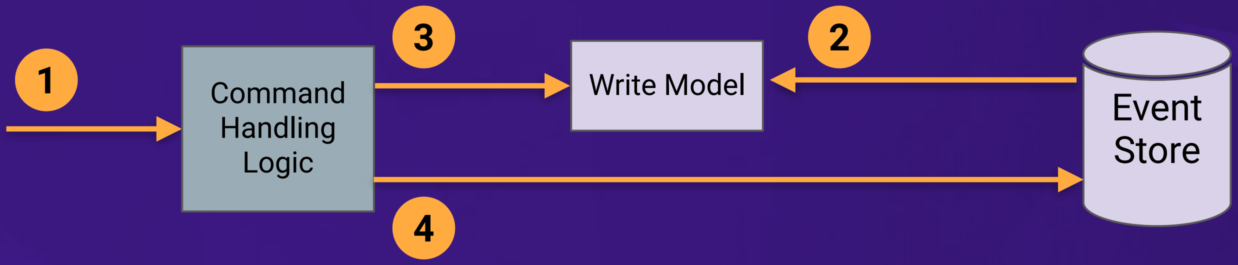
1. A write operation is initiated (Command)
2. State of the Write Model is Sourced

Combining CQRS with ES



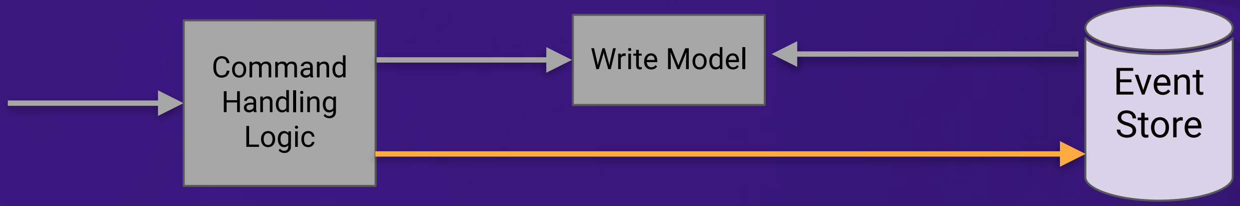
1. A write operation is initiated (Command)
2. State of the Write Model is Sourced
3. Command Handling Logic inspects the state and prepares the changes

Combining CQRS with ES

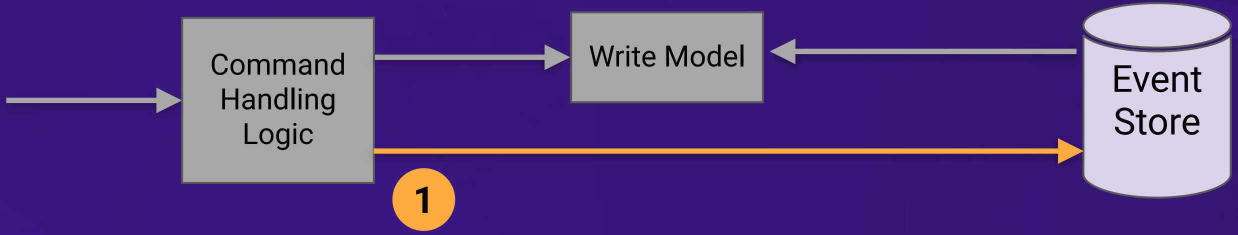


1. A write operation is initiated (Command)
2. State of the Write Model is Sourced
3. Command Handling Logic inspects the state and prepares the changes
4. The changes are reflected as Events, written to the Event Store

Projections but on Steroids

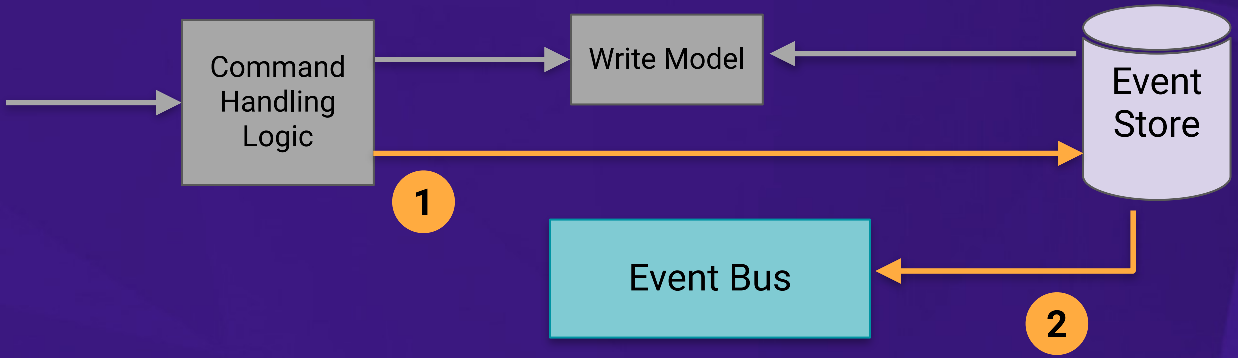


Projections but on Steroids



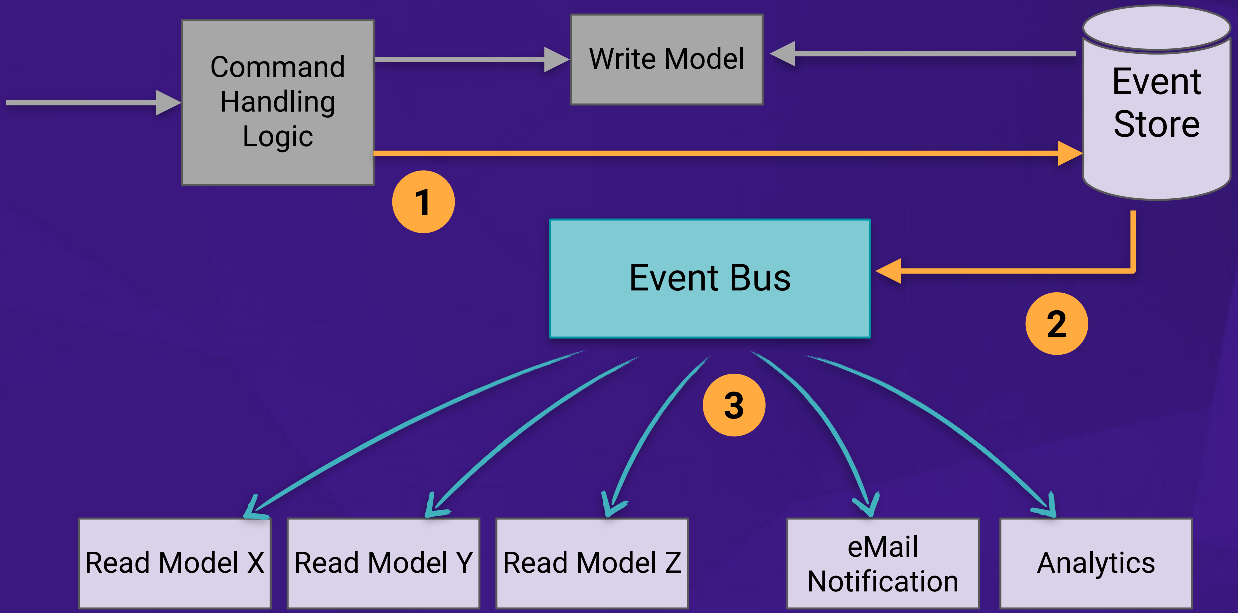
1. Generated events are stored as previously described

Projections but on Steroids



1. Generated events are stored as previously described
2. Once successfully stored, these Events are emitted through an Event Bus

Projections but on Steroids



1. Generated events are stored as previously described
2. Once successfully stored, these Events are emitted through an Event Bus
3. Many projections may be derived.
   In fact, not only projections bot other types of Event consumers

Enough talking, let's look at an

Example

and how PDFs can be incorporated

Requirements

- Creation of invoices
  Invoices are initially in a draft state
- Preview-PDFs of an invoice draft may be generated at any time.
  Generated preview-PDFs must be archived and accessible
- With the finalization of an invoice, a finalized PDF is generated
- Finalized invoices are made available through the customer portal
- If required, an invoice may be revoked (changing the state to draft again).

The Example

- Application for creating, editing and tracking invoices
- An invoice might be edited by multiple persons (adding positions, descriptions)
- Recipients may see their invoices through a portal

The Board

Create Preview PDF

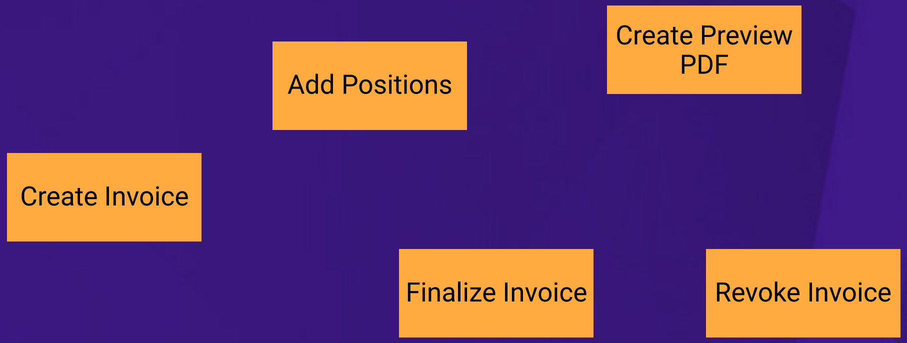
Add Positions

Create Invoice

Finalize Invoice

Revoke Invoice

The Board



Legend: Command

The Board

Create Invoice

Add Positions

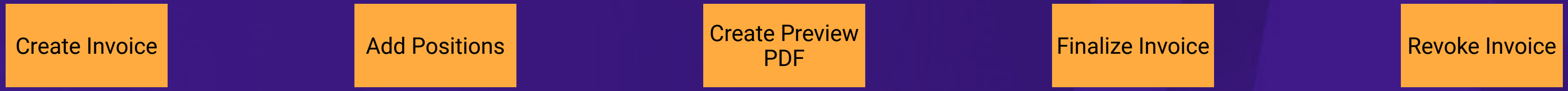
Create Preview PDF

Finalize Invoice

Revoke Invoice

Legend: Command

The Board



Legend: Command Event

The Board

Create Invoice

Add Positions

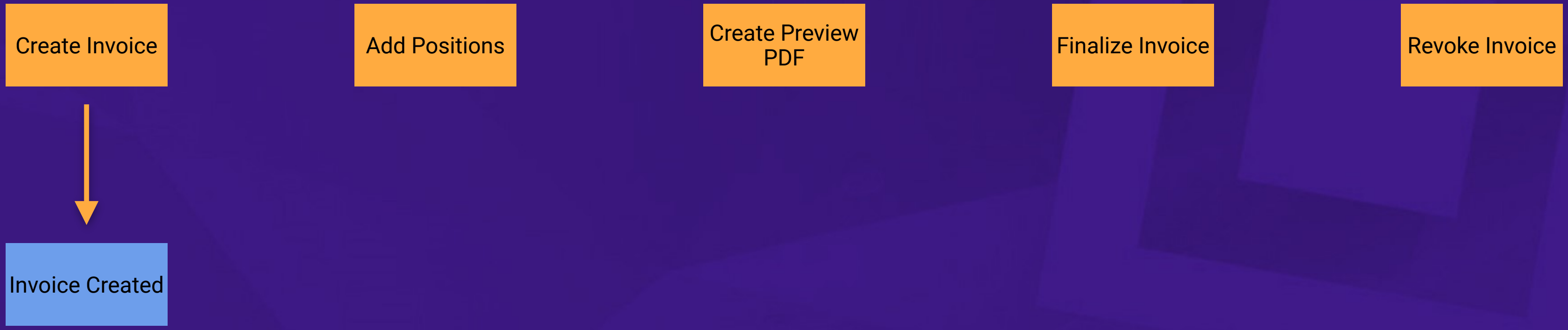
Create Preview PDF

Finalize Invoice

Revoke Invoice

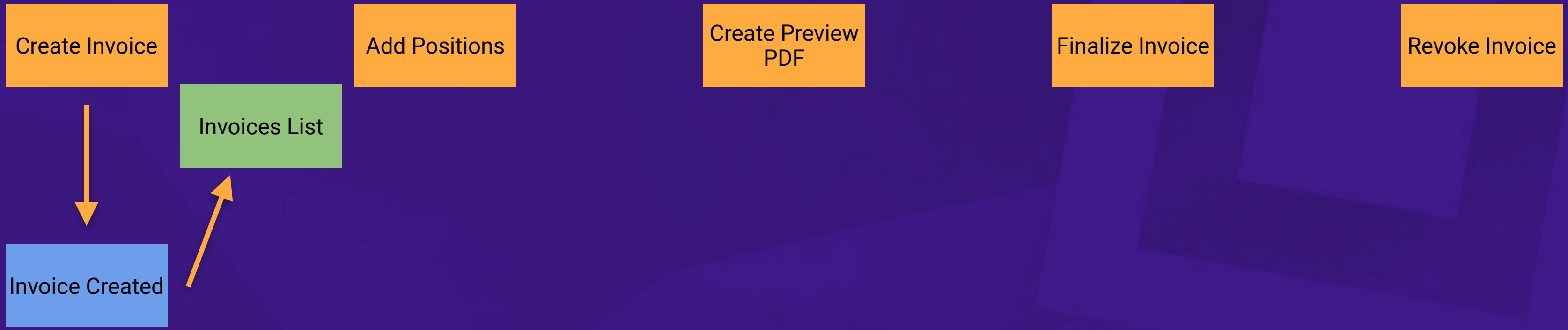
Legend: Command Event View

The Board



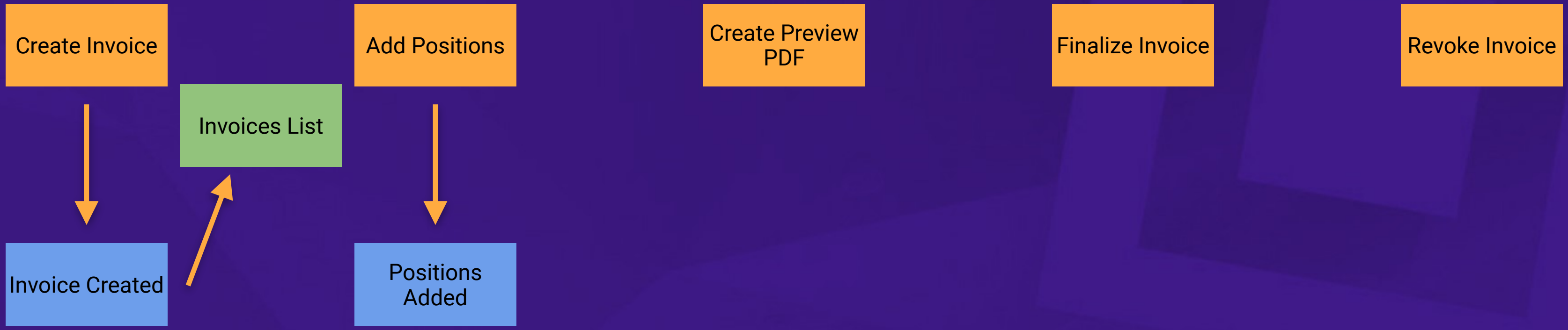
Legend: Command Event View

The Board



Legend: 🟧 Command 🟦 Event 🟩 View

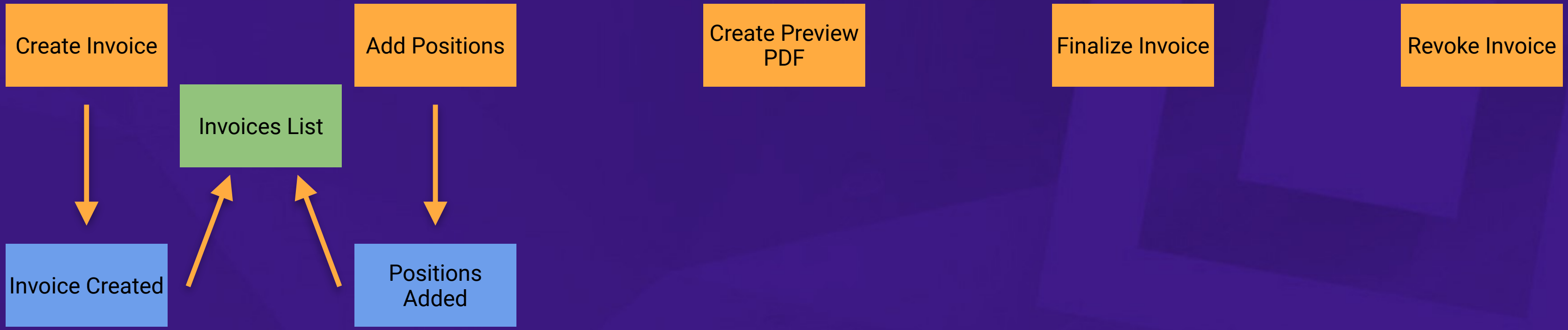
The Board



Legend: Command Event View

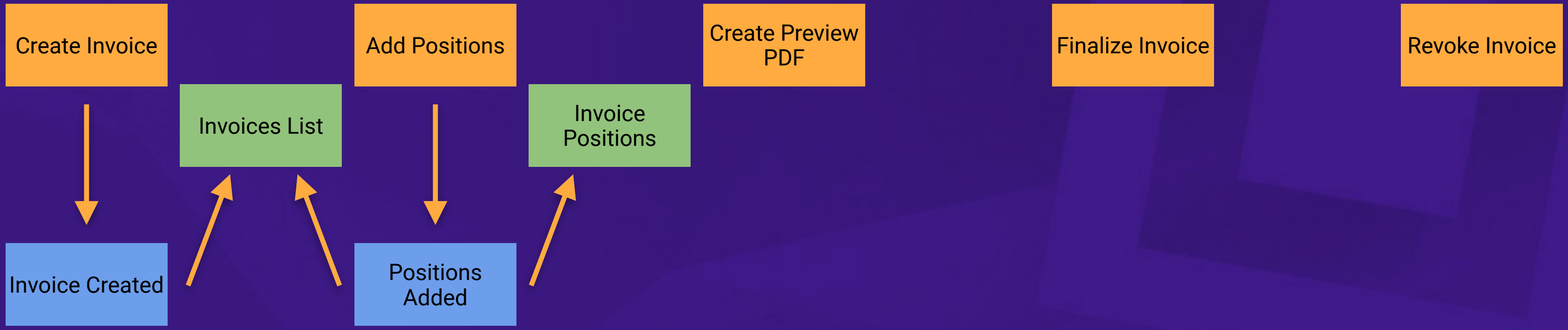
digital frontiers

The Board



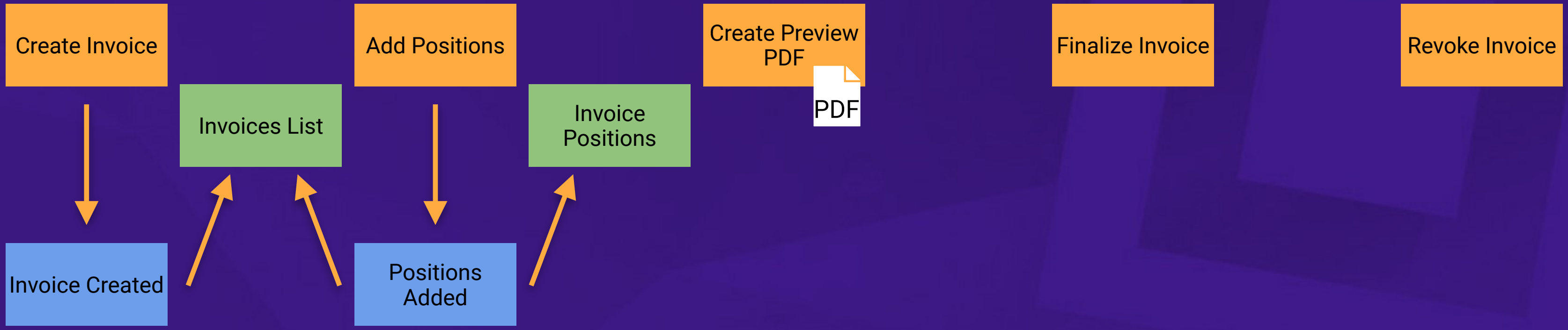
Legend: 🟧 Command 🟦 Event 🟩 View

The Board



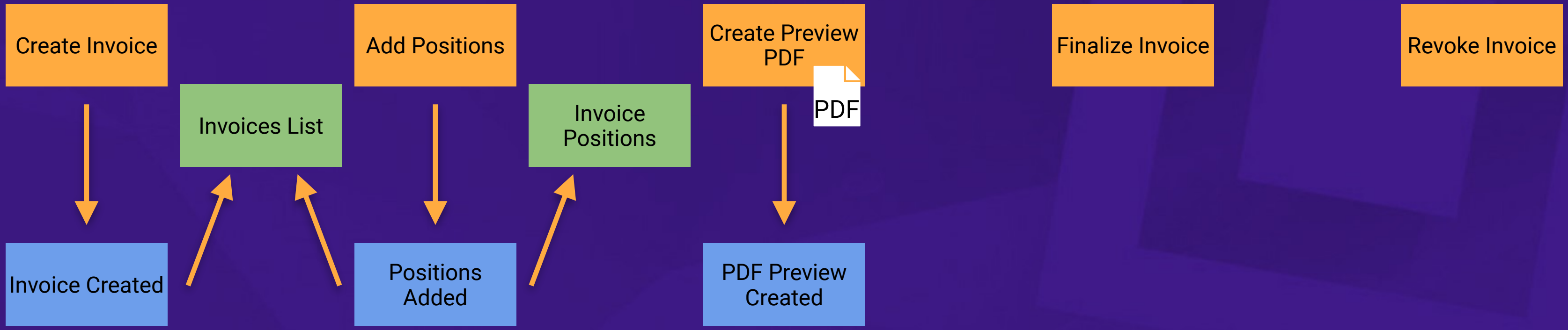
Legend: Command Event View

The Board



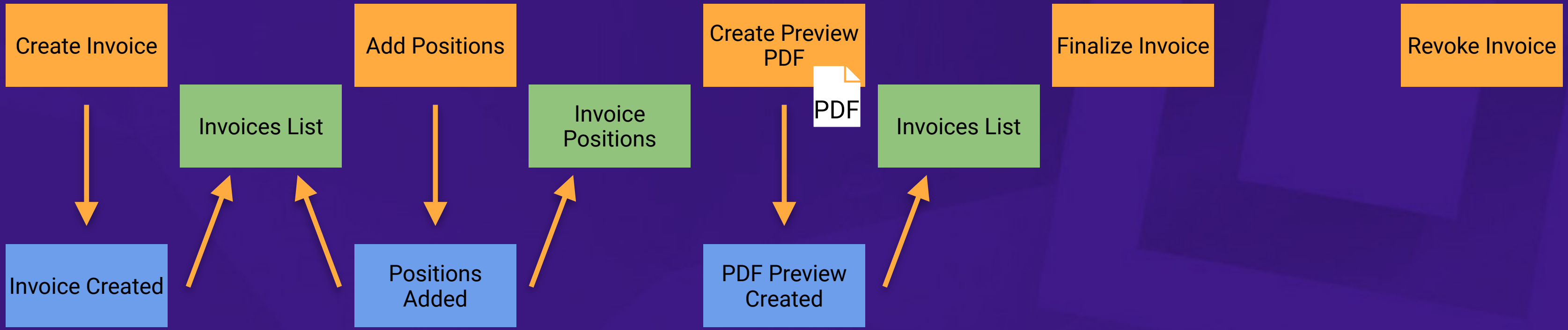
Legend: 🟧 Command 🟦 Event 🟩 View

The Board



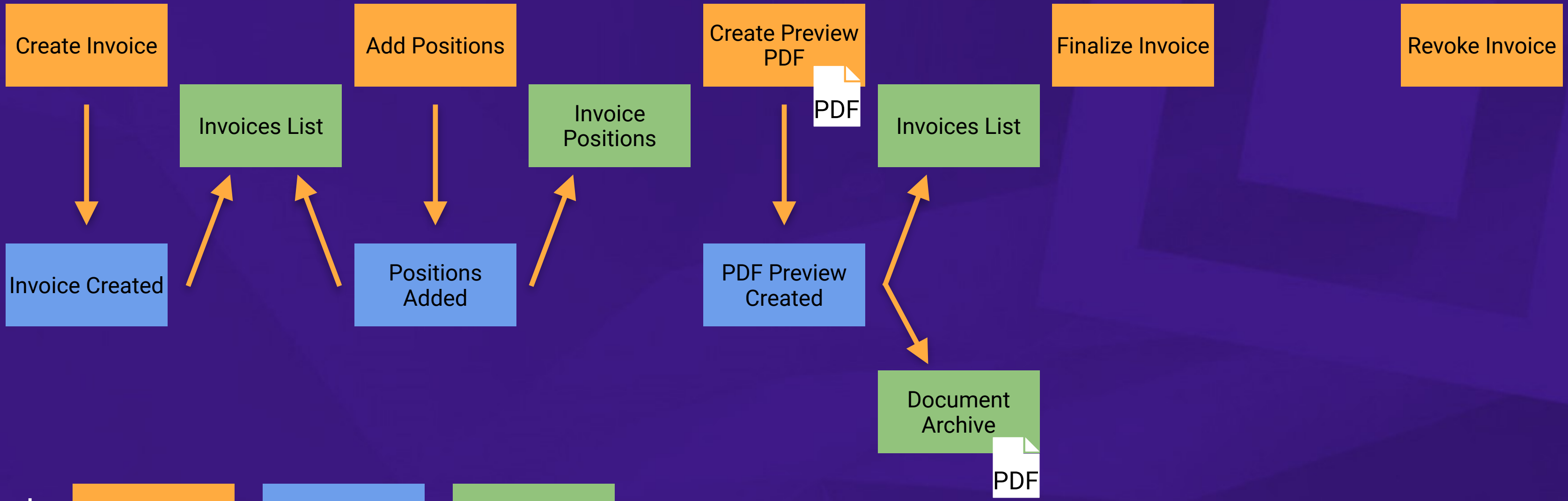
Legend: 🟧 Command 🟦 Event 🟩 View

The Board



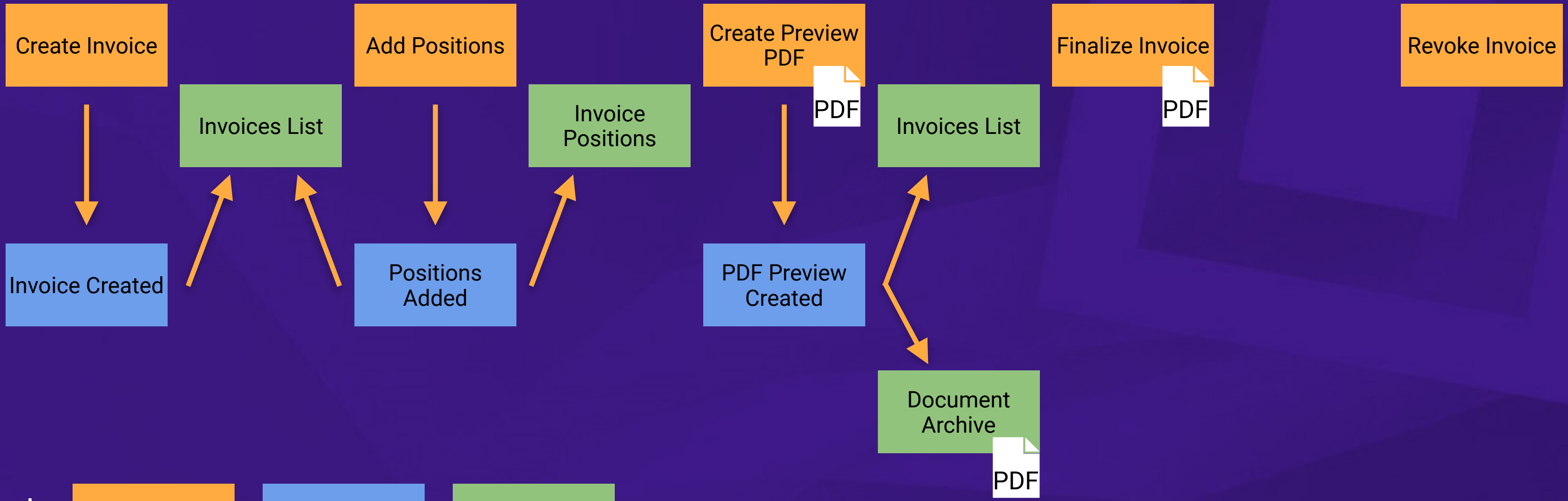
Legend: 🟧 Command 🟦 Event 🟩 View

The Board



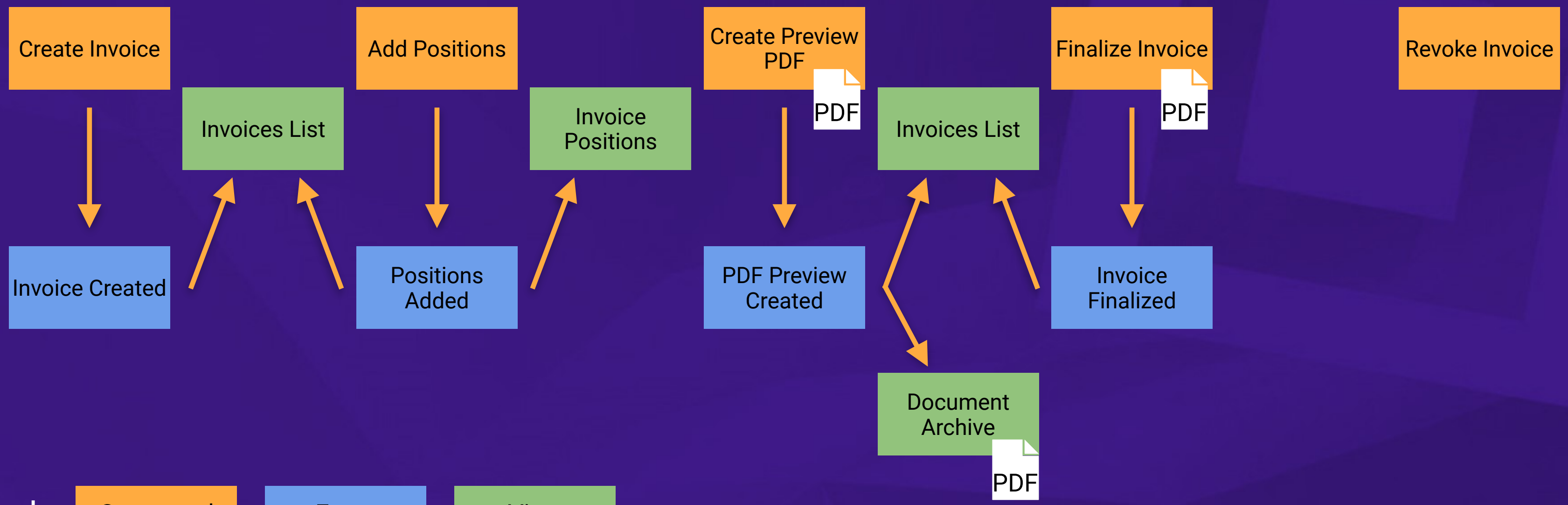
Legend: Command Event View

The Board



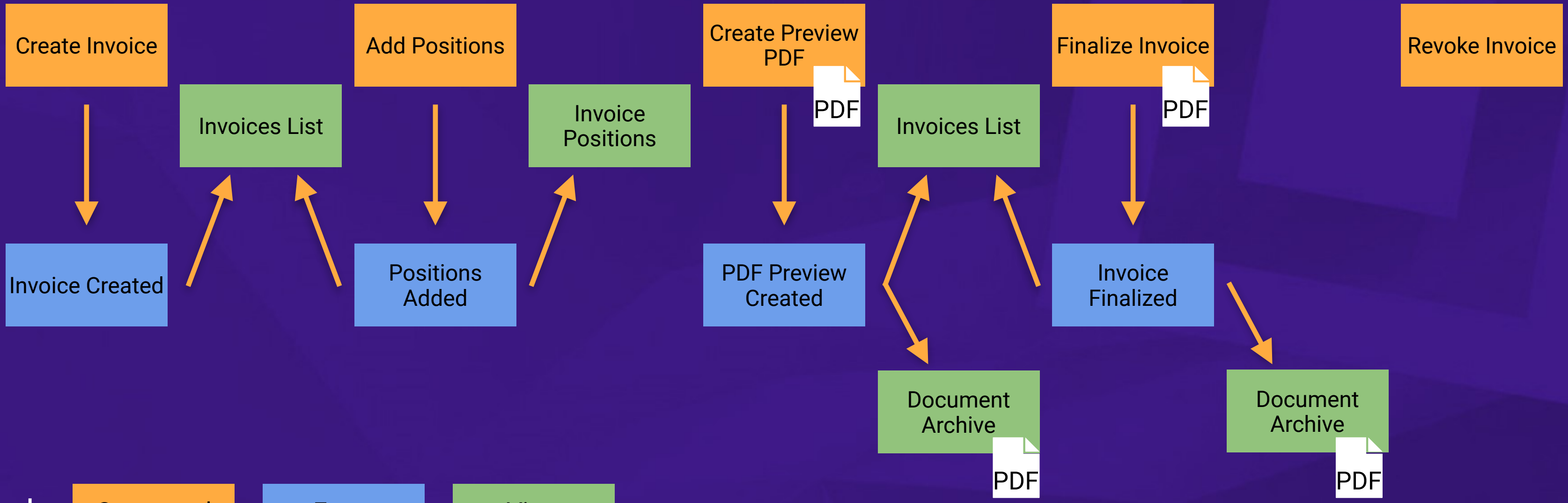
Legend: 🟧 Command 🟦 Event 🟩 View

The Board



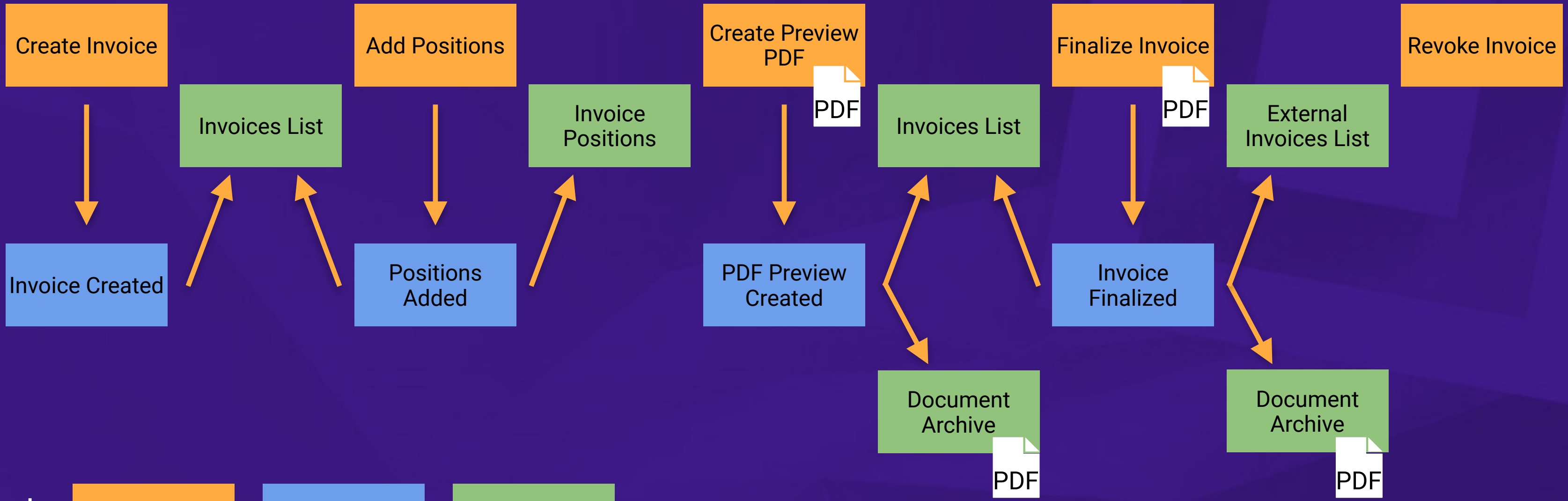
Legend: Command Event View

The Board



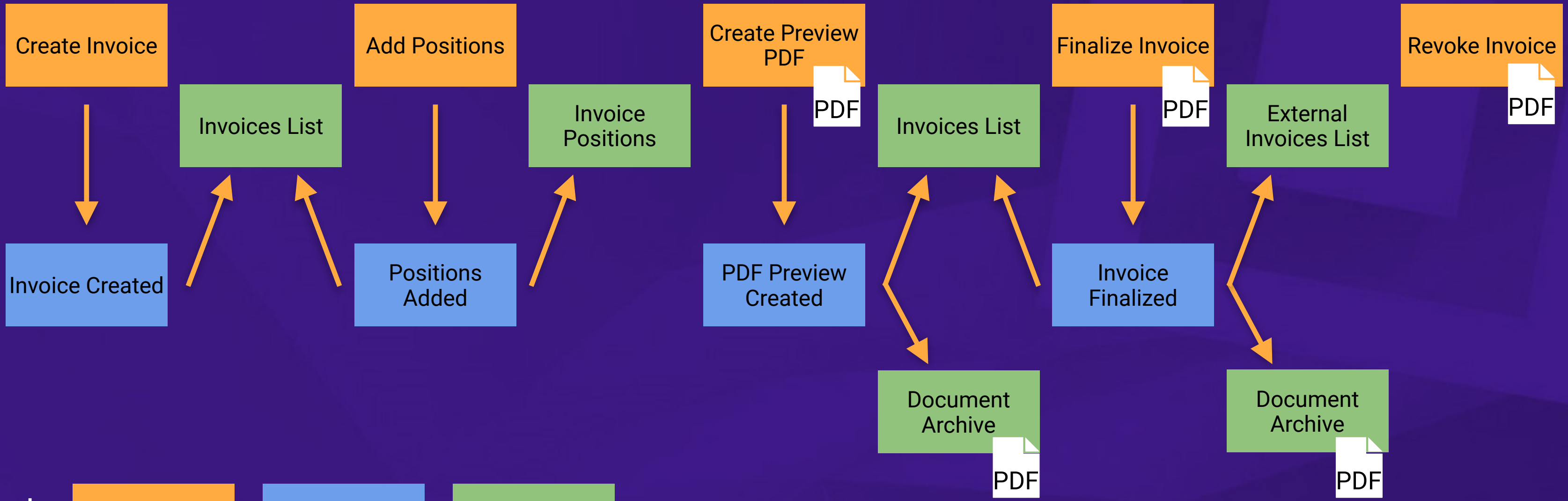
Legend: 🟧 Command 🟦 Event 🟩 View

The Board



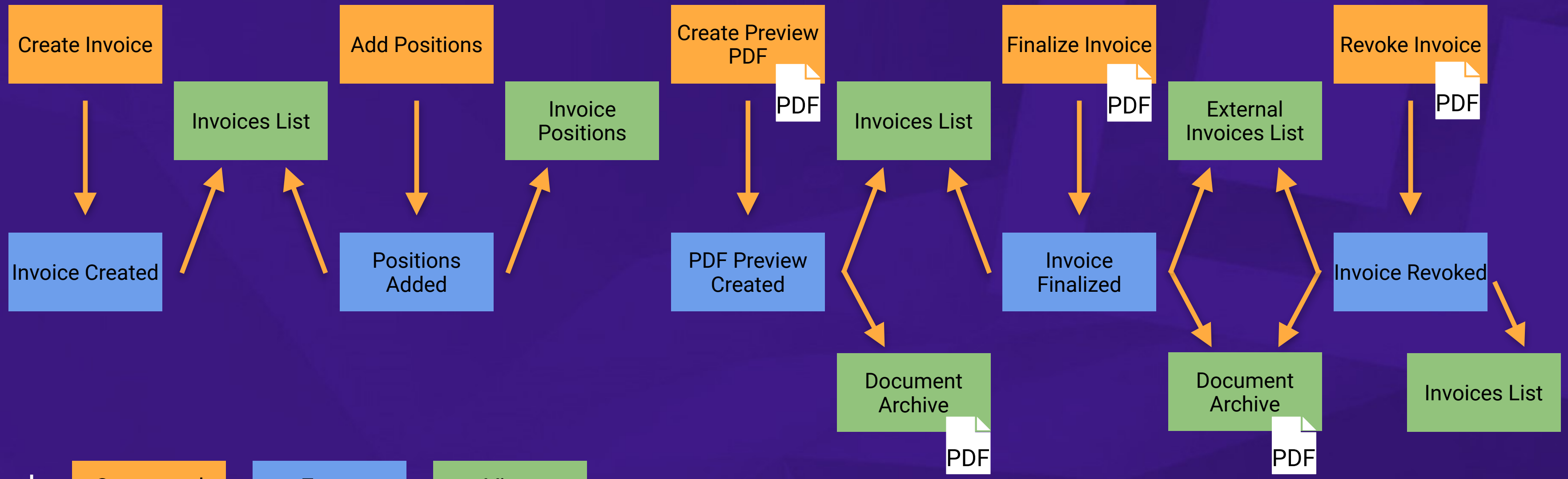
Legend: 🟧 Command 🟦 Event 🟩 View

The Board



Legend: 🟧 Command 🟦 Event 🟩 View

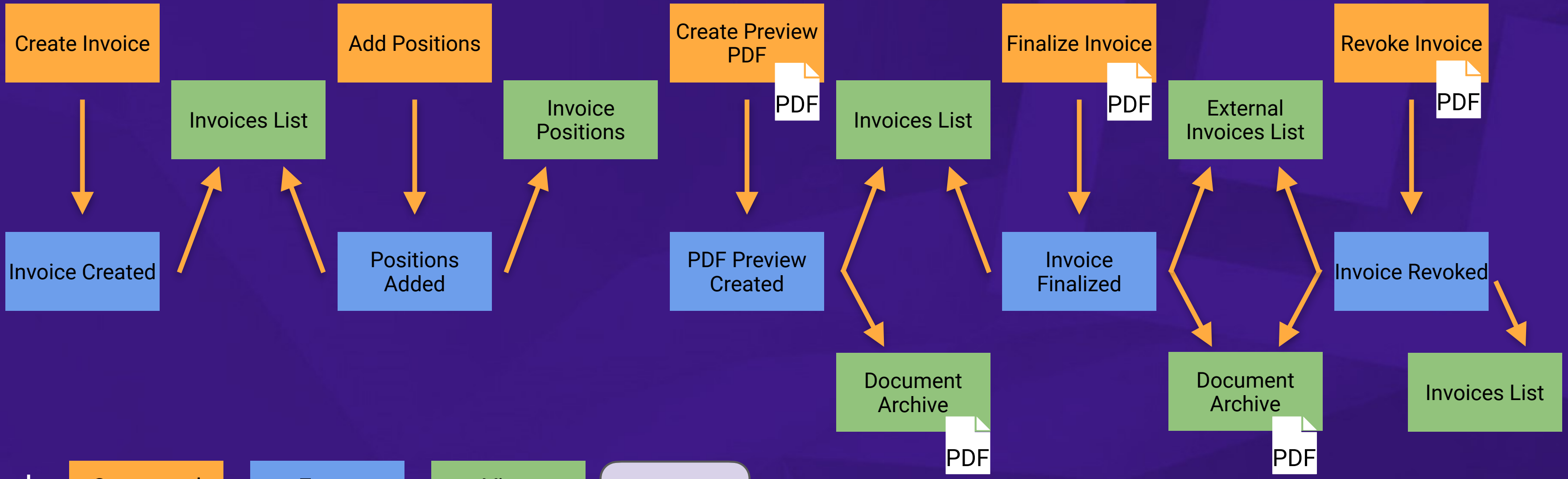
The Board



Legend: 🟧 Command 🟦 Event 🟩 View

digital frontiers

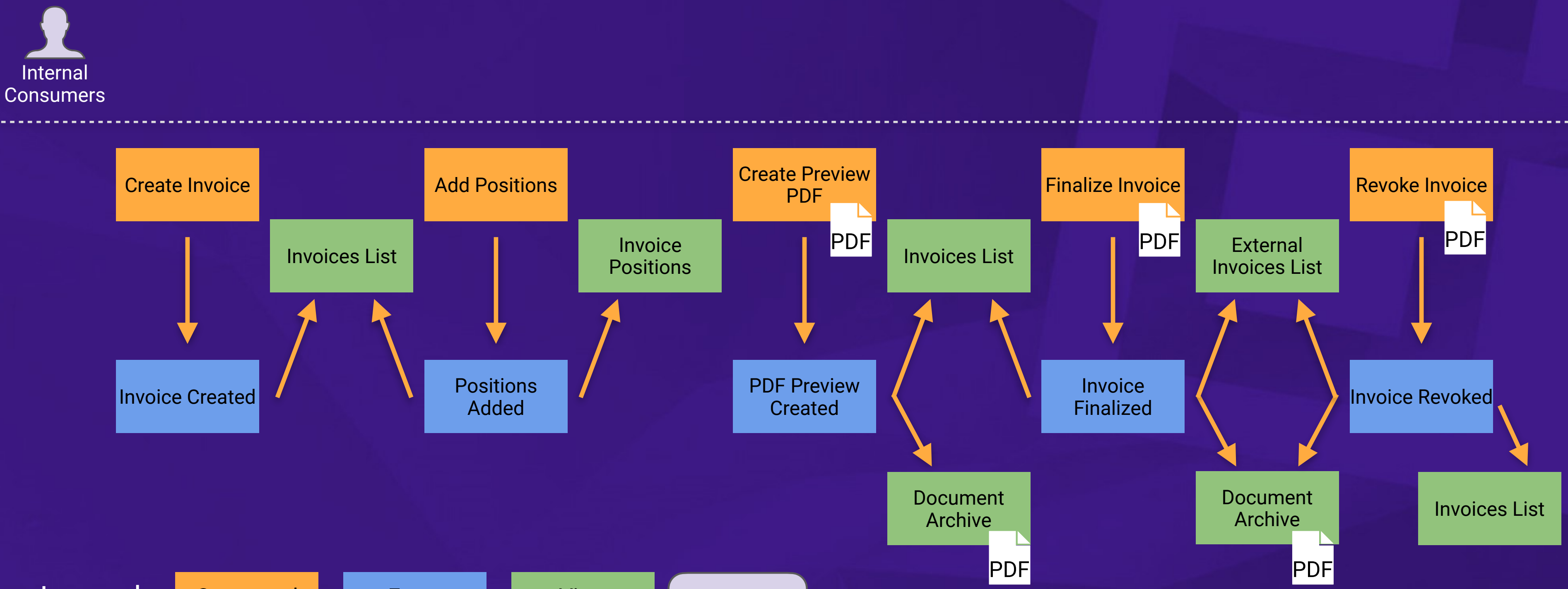
The Board



Legend: 🟧 Command 🟦 Event 🟩 View ⬜ UI

digital frontiers

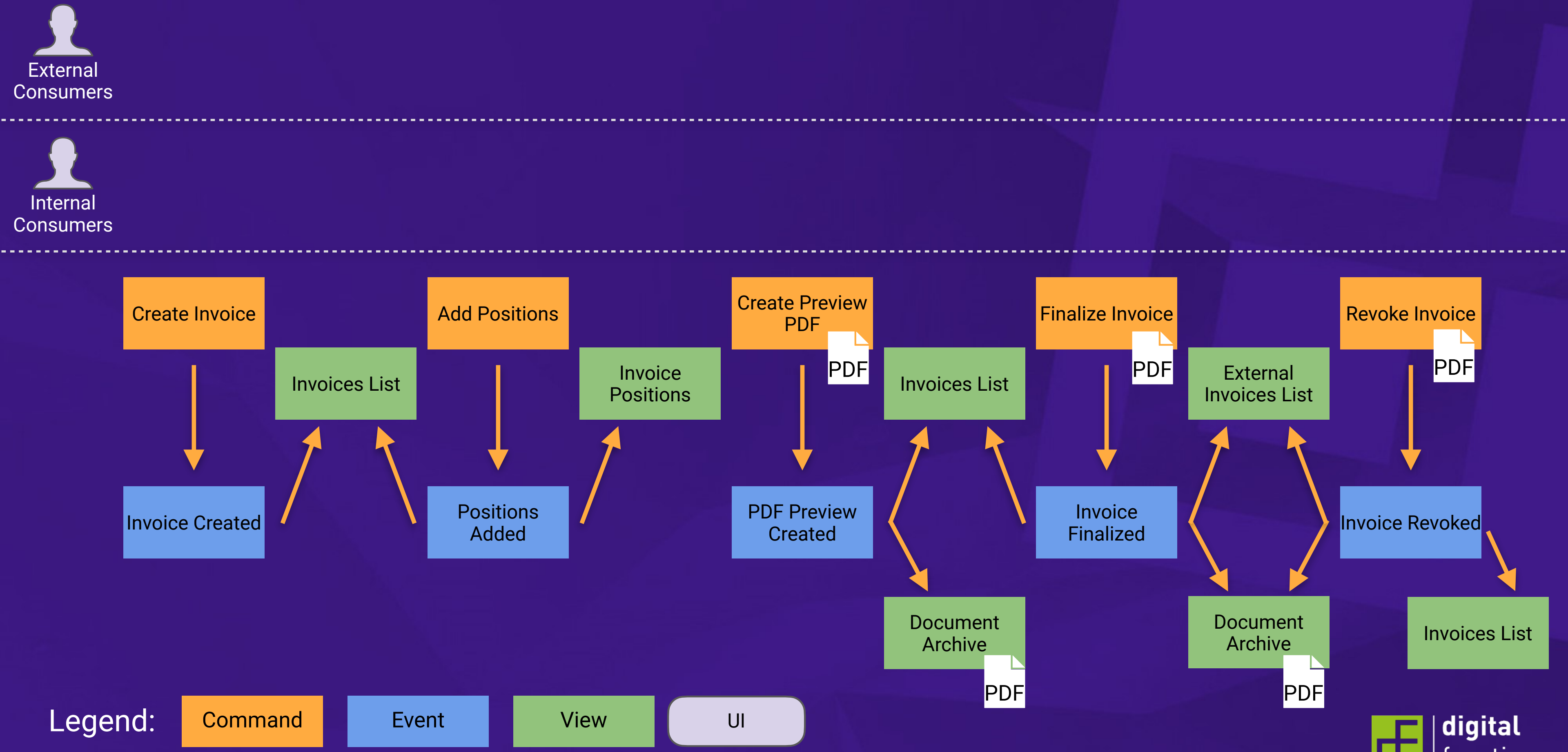
The Board



Legend: Command Event View UI

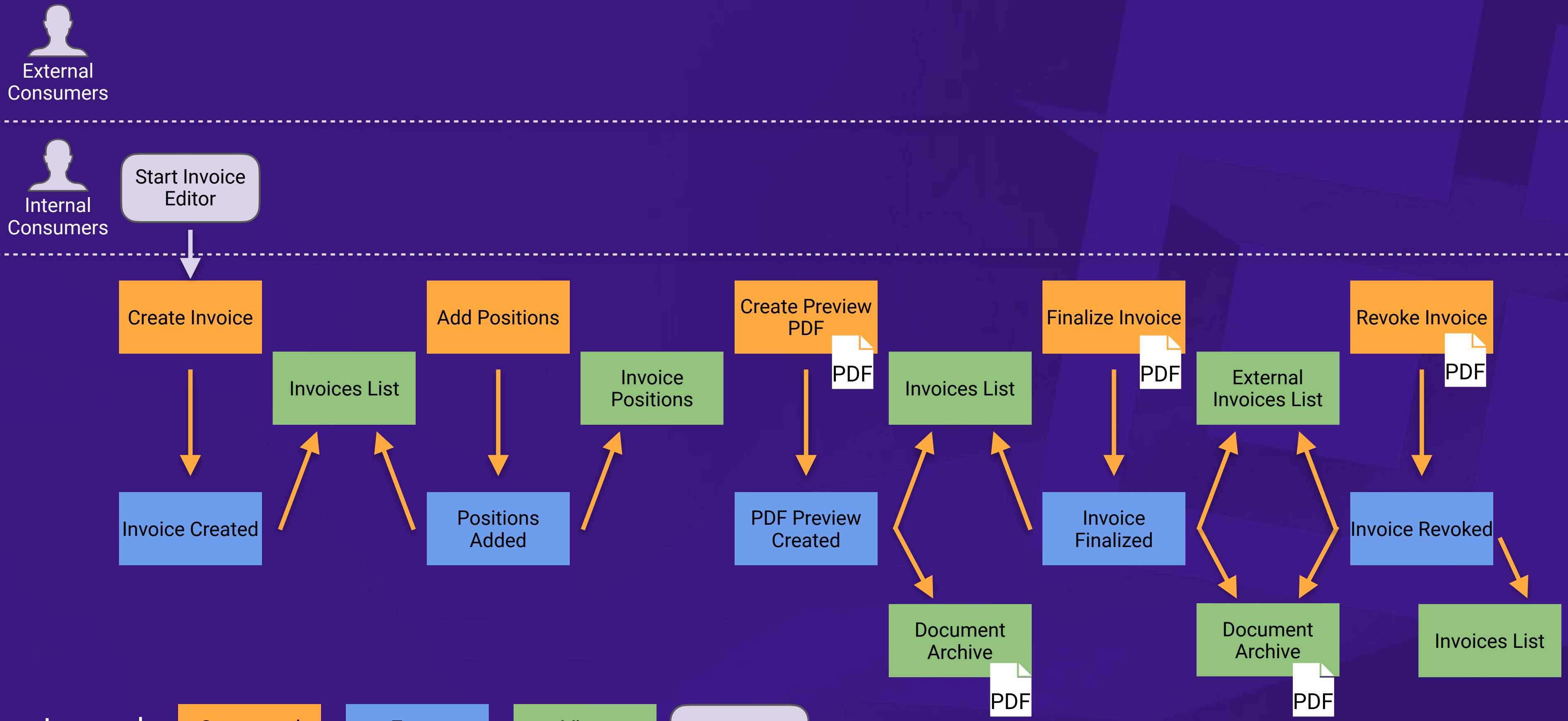
digital frontiers

The Board



digital frontiers

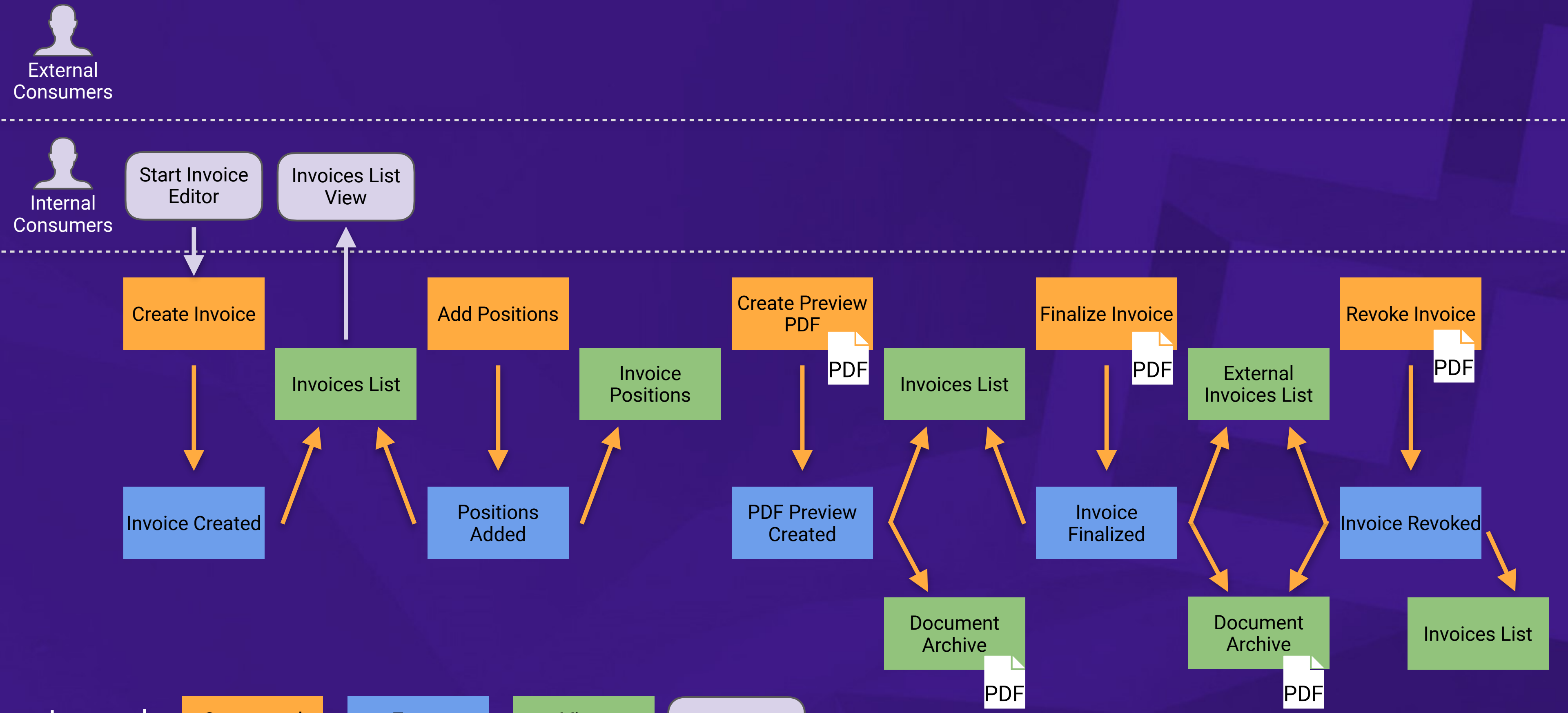
The Board



Legend: 🟧 Command 🟦 Event 🟩 View ⬜ UI

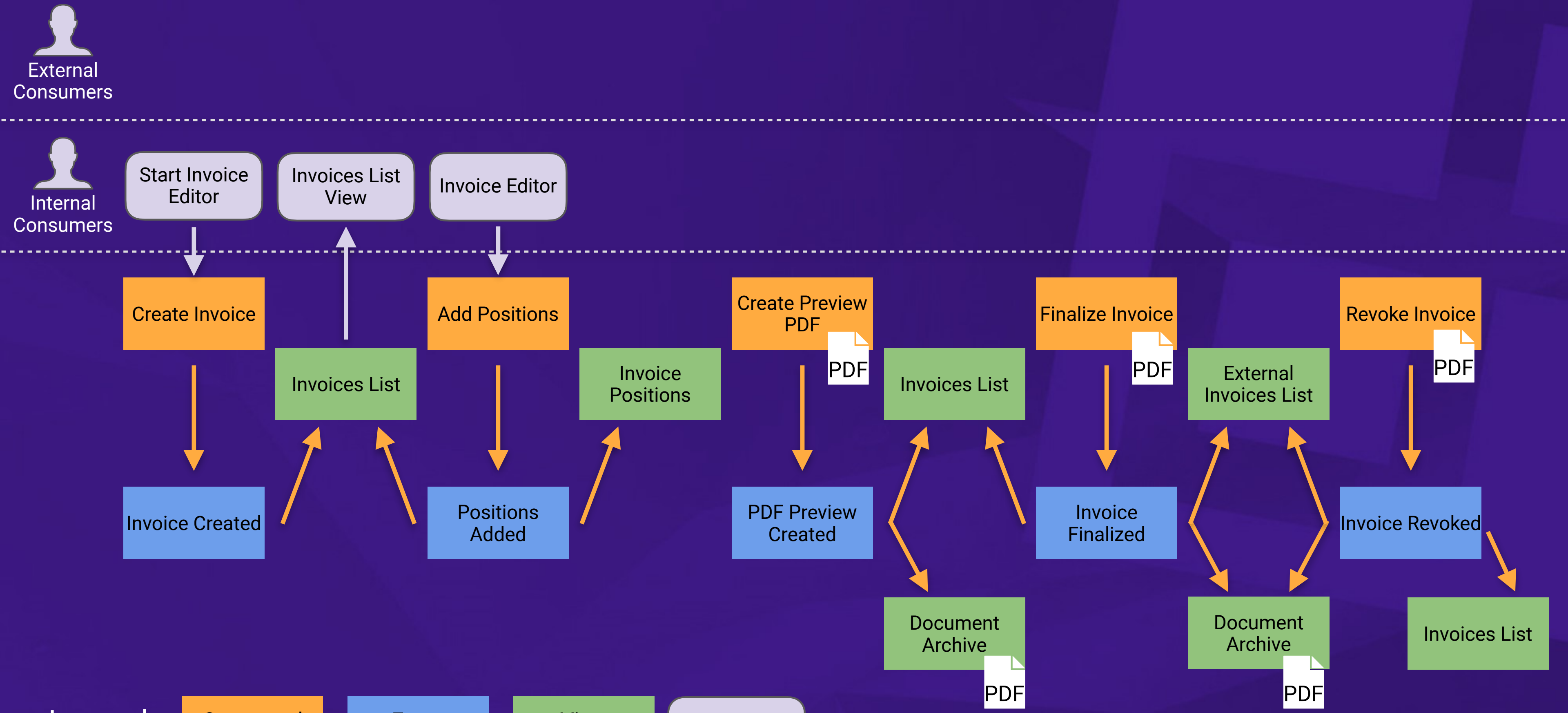
digital frontiers

The Board



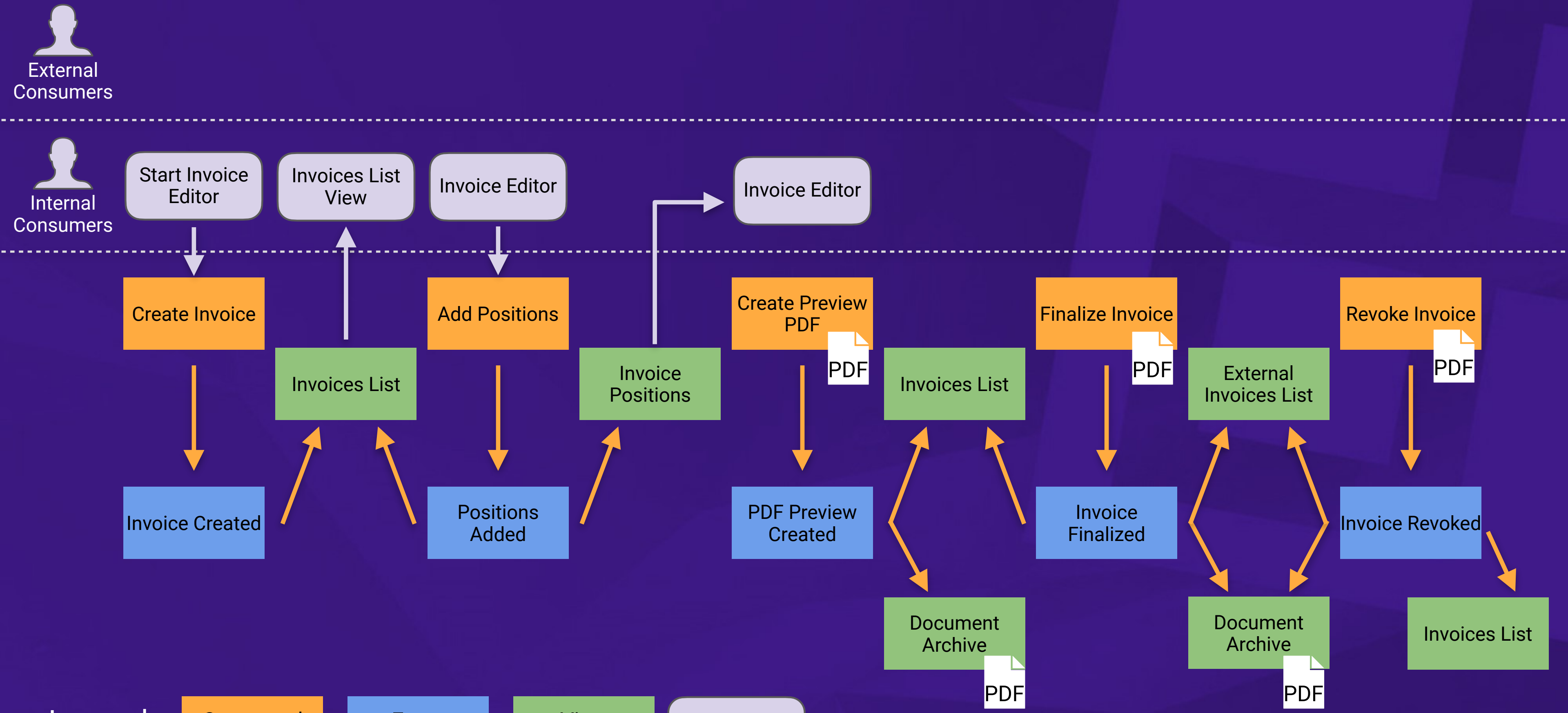
Legend: 🟧 Command 🟦 Event 🟩 View ⬜ UI

The Board



Legend: 🟧 Command 🟦 Event 🟩 View ⬜ UI

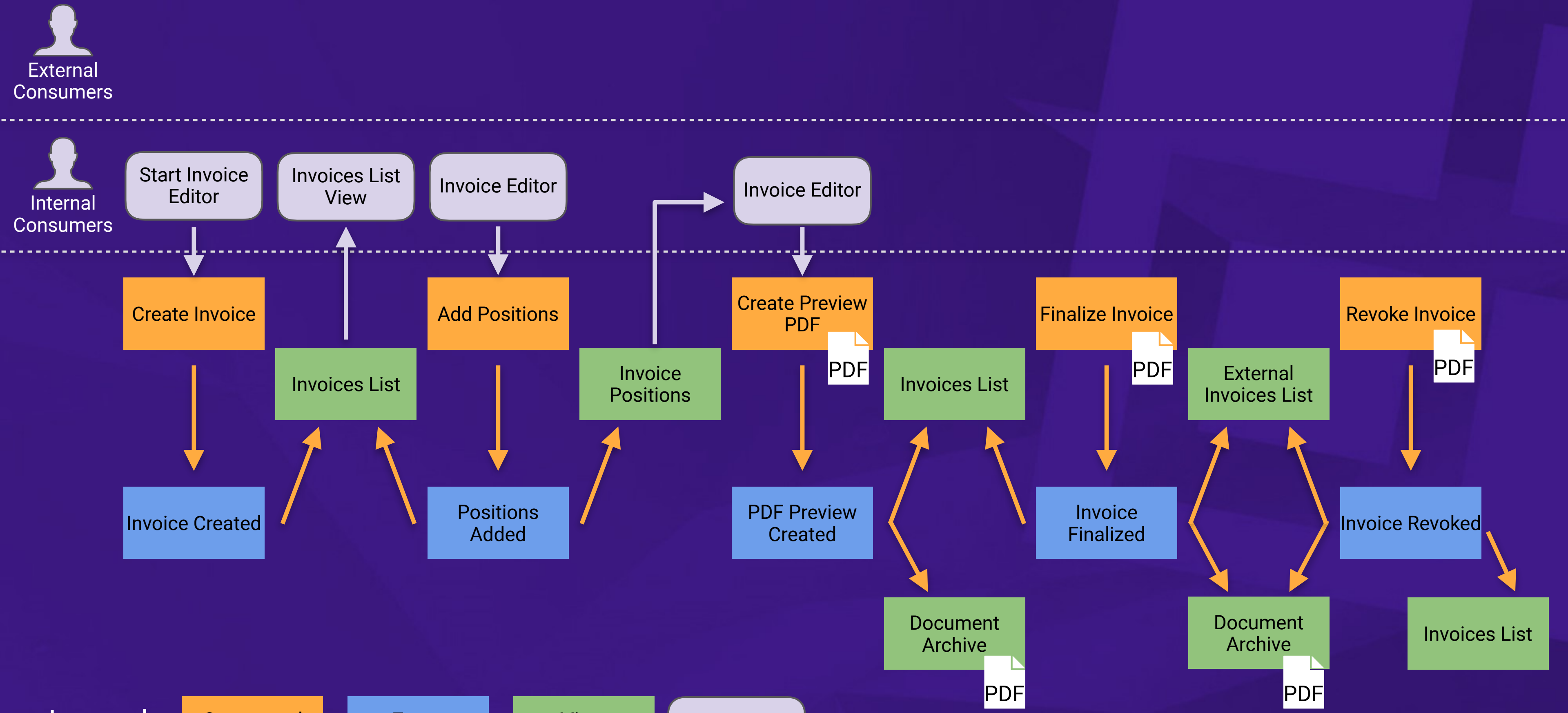
The Board



Legend: 🟧 Command 🟦 Event 🟩 View ⬜ UI

digital frontiers

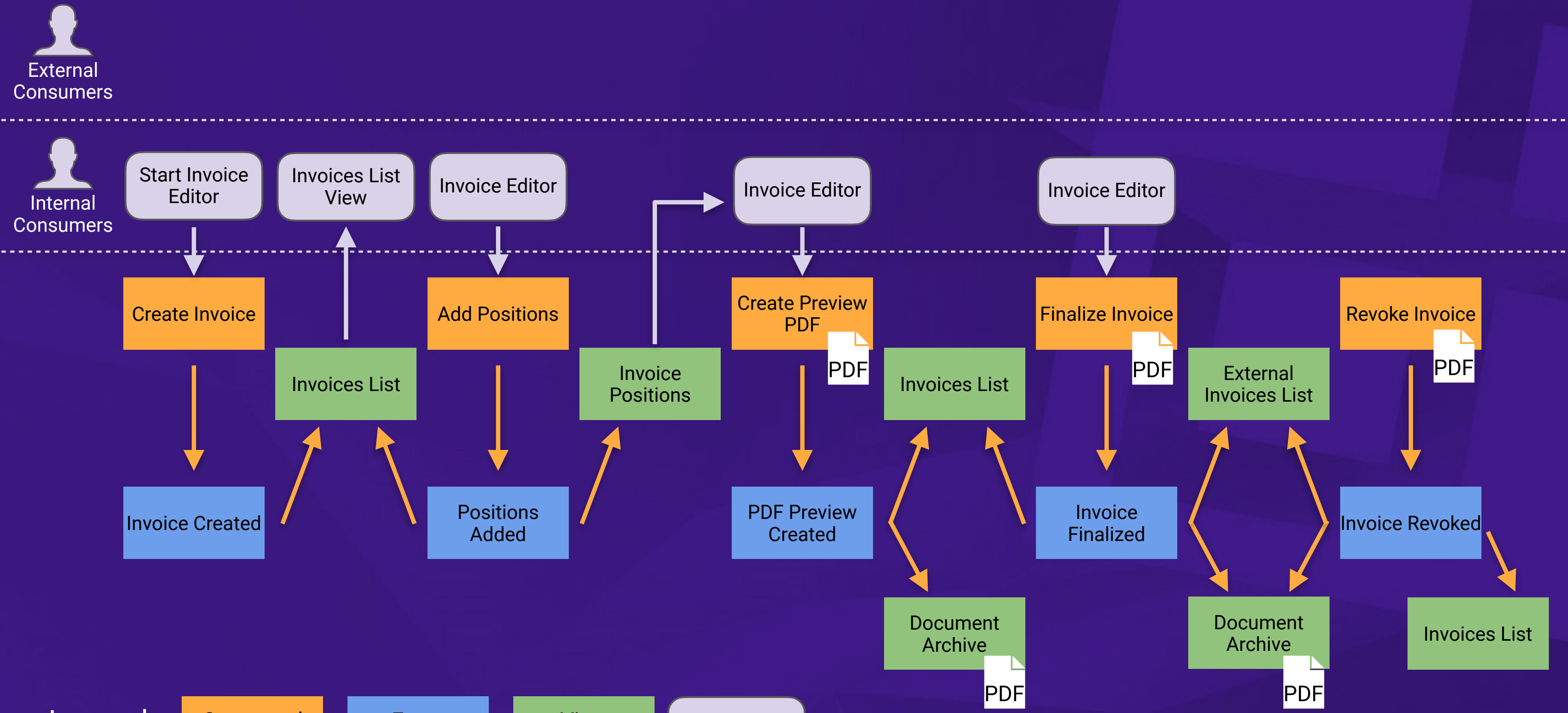
The Board



Legend: 🟧 Command 🟦 Event 🟩 View ▢ UI

digital frontiers

The Board

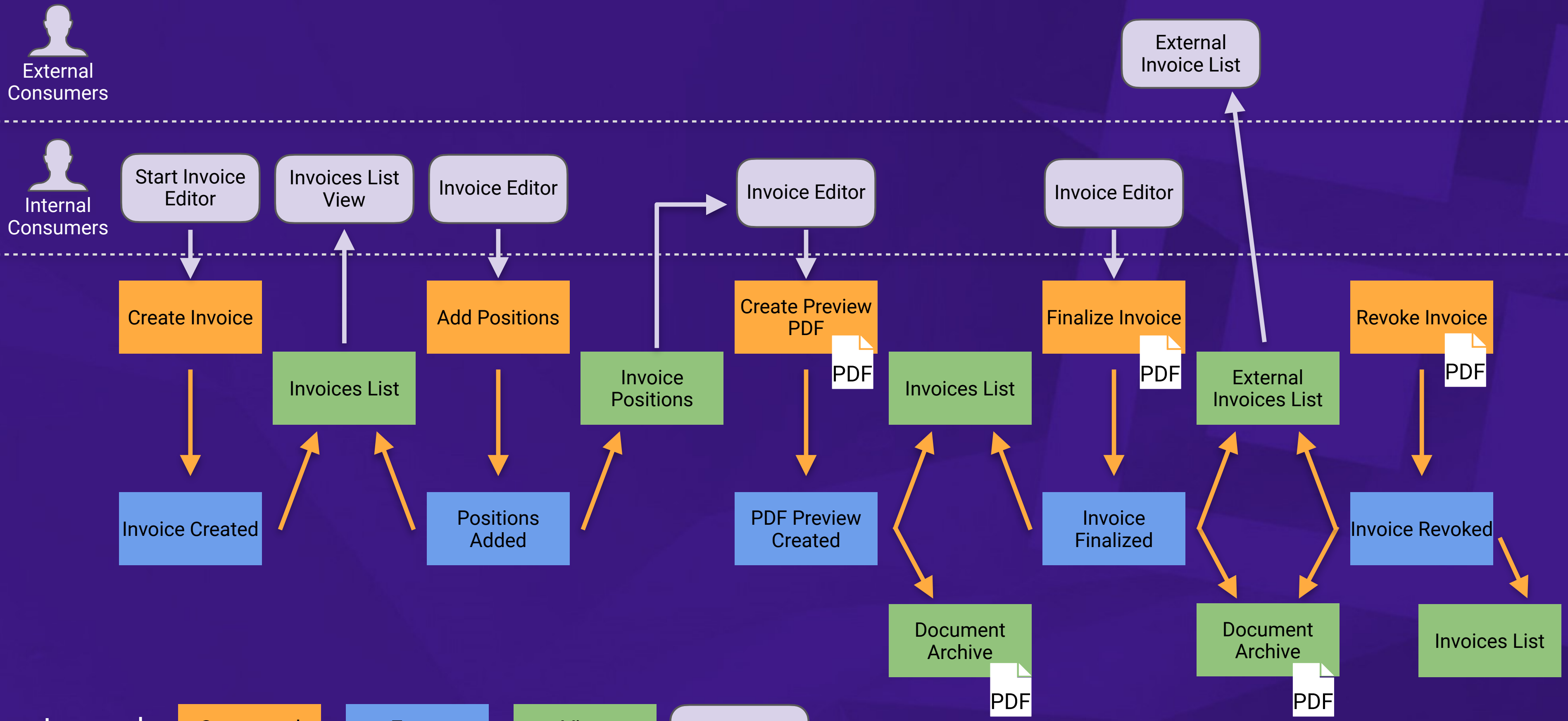


Legend:

🟧 Command	🟦 Event	🟩 View	UI

digital frontiers

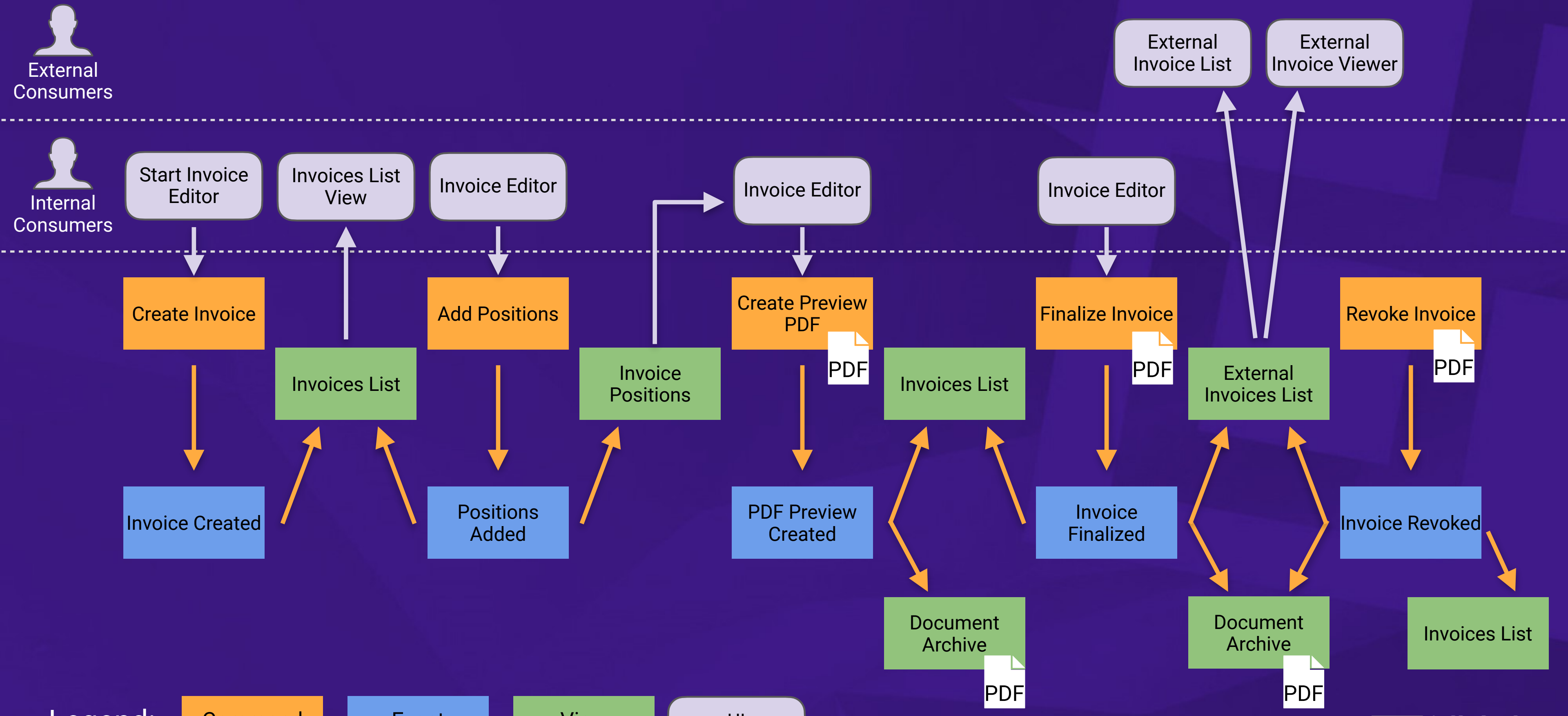
The Board



Legend: 🟧 Command 🟦 Event 🟩 View ⬜ UI

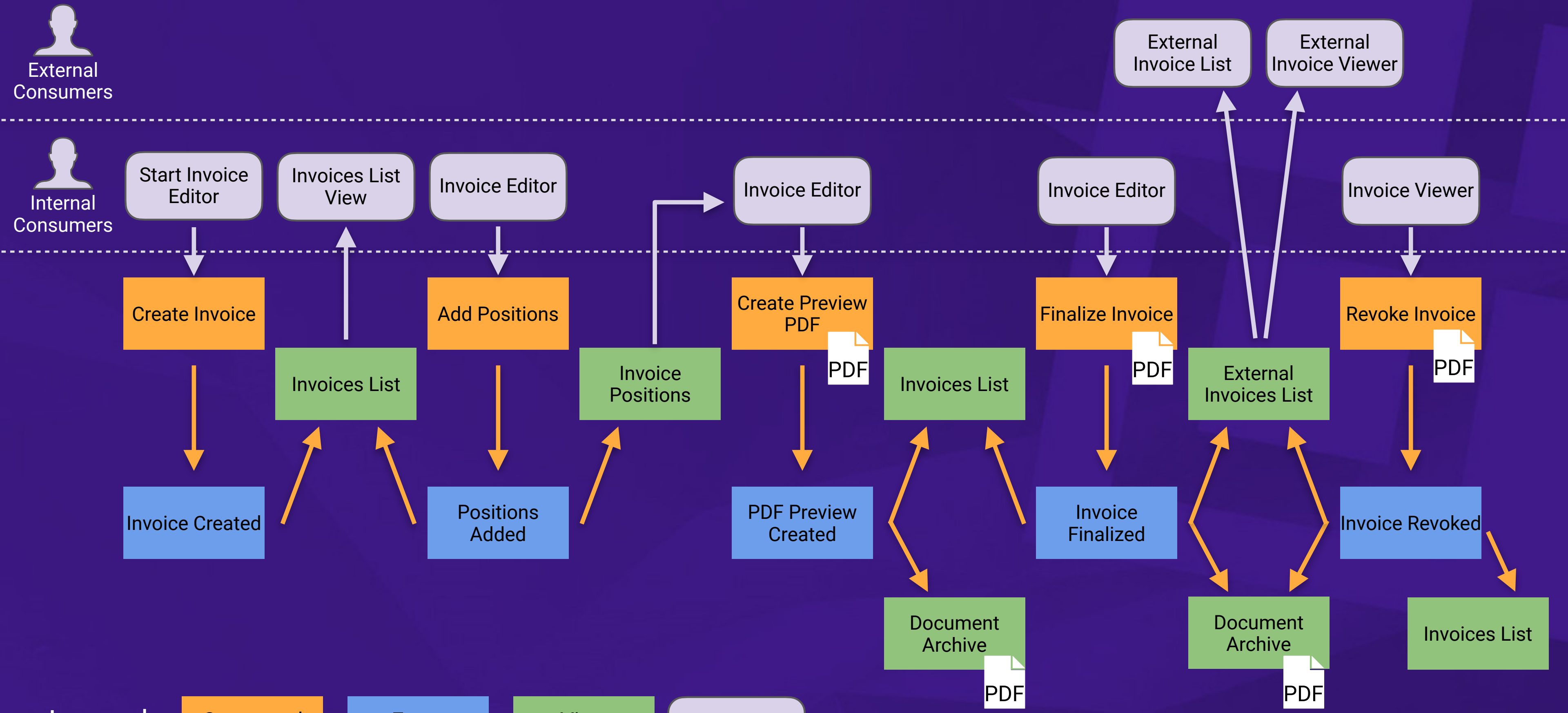
digital frontiers

The Board



Legend: Command Event View UI

The Board



Legend: Command Event View UI

Thank you for listening! Any Questions?

Photo by Matt Walsh on Unsplash

François Fernandès
Senior Solution Architect

@tellme_francois
github.com/fernanfs

https://blog.digitalfrontiers.de
https://www.digitalfrontiers.de

digital frontiers