

The Multiplying Architecture

Eder Ignatowicz

Principal Software Engineer

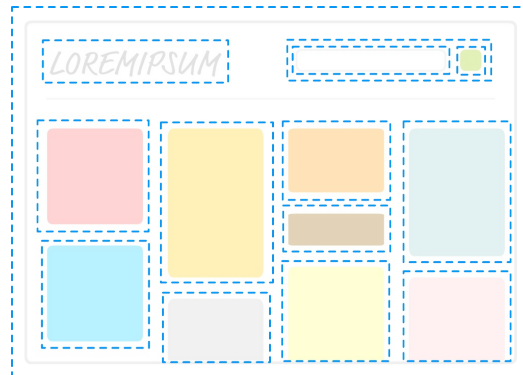
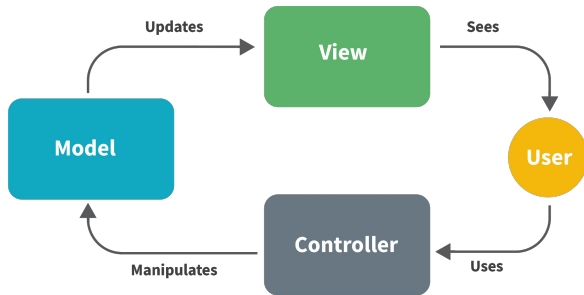
@ederign



Landscape

We live in a world where **web technologies** have dominated software development.

Default choice for most applications.



Foundation

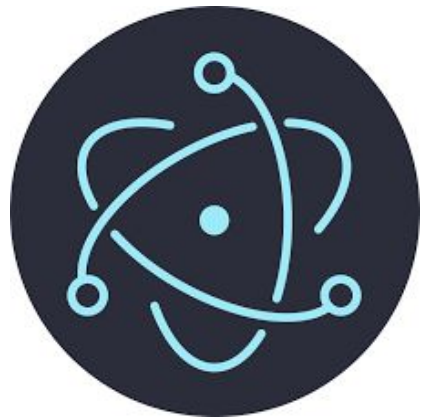
Well thought and understood set of Standards, Patterns and Techniques as a strong foundation.



TypeScript

Static Typed Language

TypeScript created the perfect compromise for Static Type Languages believers.



Browser Everywhere

Browser is now more than just the window for the internet.

Browsers became part of an important trend as the mechanism to distribute any Graphical User Interface based applications.

Architectures



Evolutionary

An evolutionary architecture supports incremental, guided change as a first principle across multiple dimensions.



Microservices

Architectural style that structures an application as a collection of independent services.



Serverless

Incorporate third-party "Backend as a Service", and/or that include custom code run as Functions.



Micro Frontends

Design approach in which a front-end app is decomposed into individual, semi-independent "microapps" working loosely together.

Why do we need a **new** architecture?

Cloud Native Tooling

requirements



Multiple Distributions

The origin of multiplying architecture is rooted in the need to distribute the same set of components in a myriad of platforms.



Minimize code changes

The components to be distributed should be preserved untouched and with avoiding feature flags.



Bridge

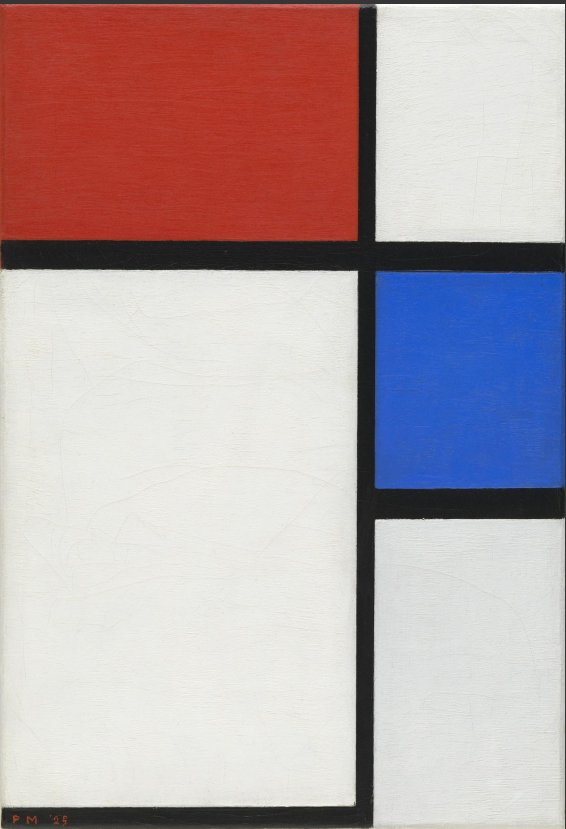
It has to embrace different generations of technology stack.

What is Software Architecture?

“Architecture is about the
important stuff.
Whatever that is.”

—
Ralph Johnson

Introducing Multiplying Architecture



What is important for the Multiplying Architecture is the ***abstraction***.

The Abstractions

core



Channel

Top level abstraction that represents the hosting environment, like a website or a desktop application.



Envelope

Enable transparent communication between Components (View/Editor) and Channel



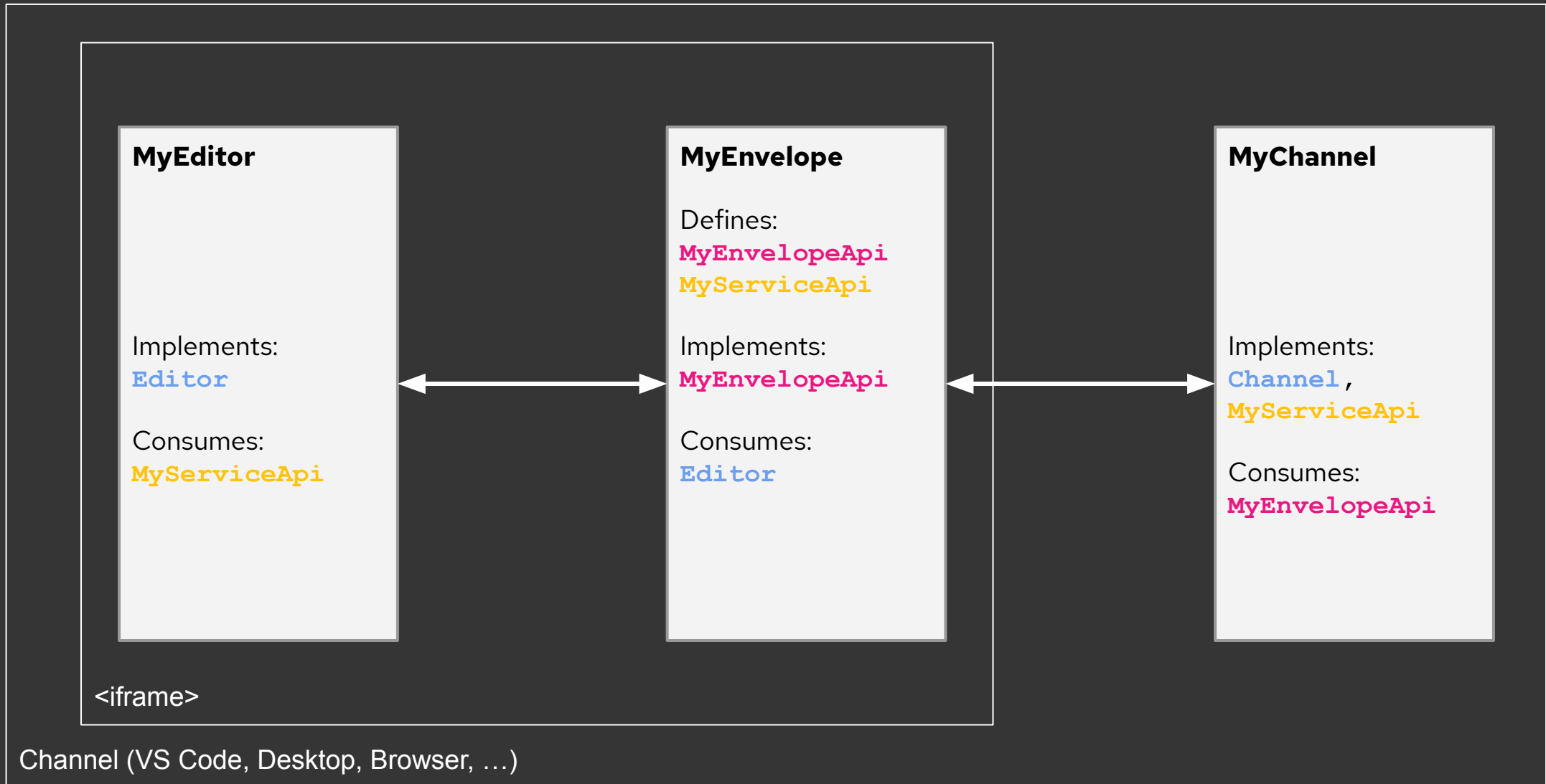
View

View is a portable set of widgets that are exposed as an unit to the Channel through the Envelope.



Editor

Editor is a specialized type of View, that gets a file content as input and is able to serve the content state back to the Channel through the Envelope.



Envelope Advantages

micro frontend



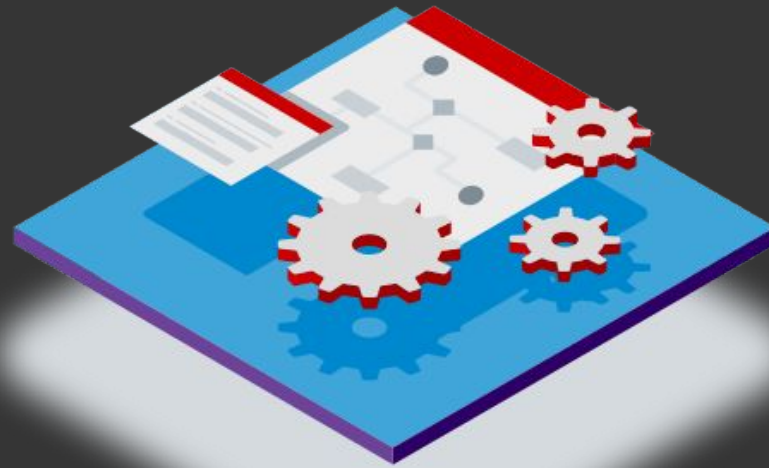
Context Isolation (CSS and JS)

Autonomous Teams

Independent Release Cycles

Type Safe Communication

Multiplying Architecture In Practice

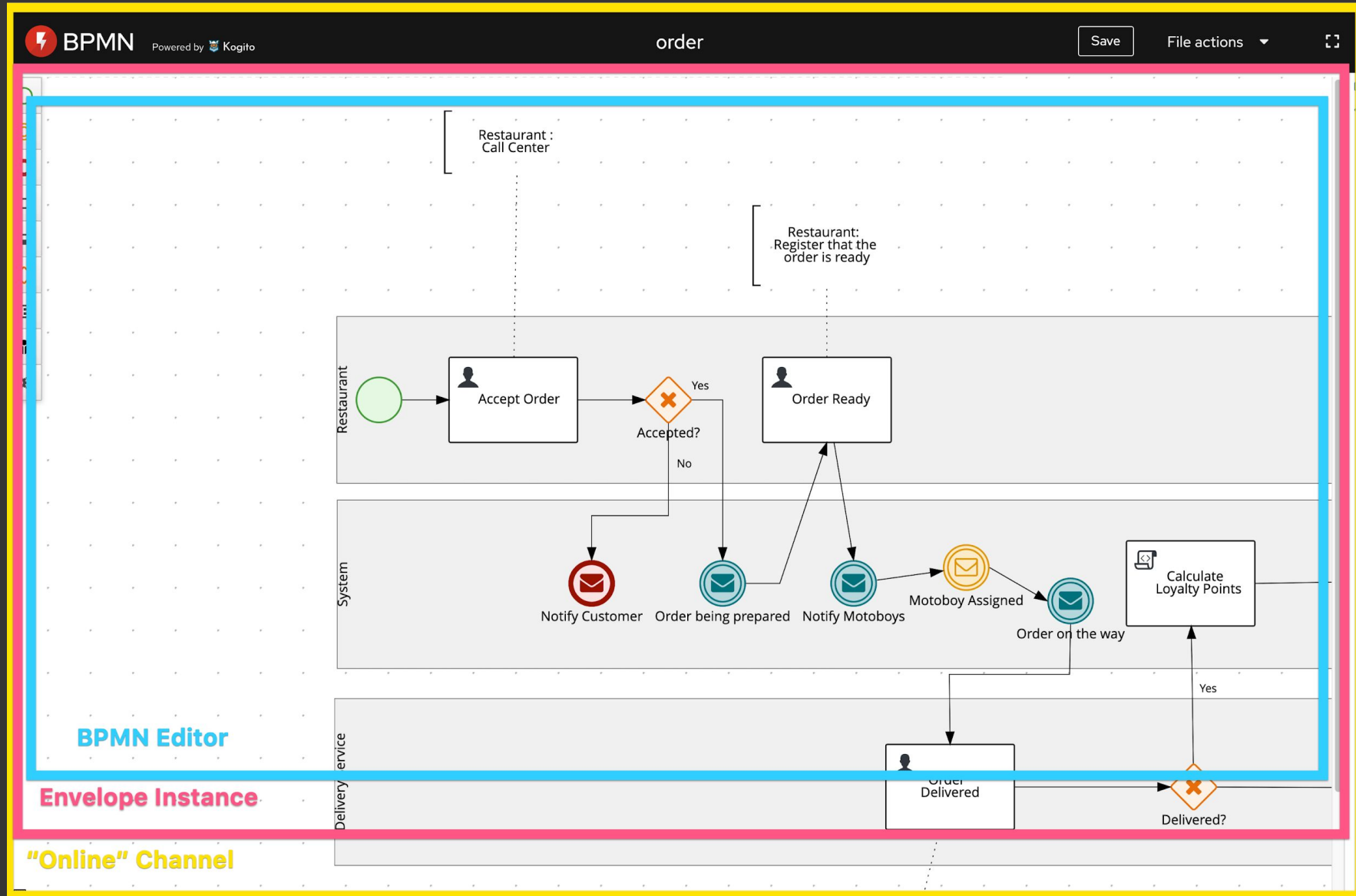


* Client side online editor

The image is a collage illustrating the KIE Tooling Channels for a client-side online editor. It features several overlapping screenshots and logos:

- Visual Studio Code:** A screenshot of a local IDE showing a BPMN diagram for 'order.bpmn2' with various tasks like 'Accept Order', 'Order Ready', and 'Calculate Supply Prices'.
- Chrome Browser:** A screenshot of a web browser displaying the 'order.bpmn2' online editor interface, showing the same BPMN diagram.
- GitHub:** A screenshot of a commit history for the file 'order.bpmn2', showing a commit by 'kmacedovarela@gmail.com' with the message 'Fixing all the variables from the bpmn2 file'.
- Read-only Visualization:** A screenshot of a 'readonly visualization' of the BPMN diagram, showing a simplified view of the process flow.
- Business Modeler Preview:** A screenshot of the 'Business Modeler Preview' window, showing a detailed view of the BPMN diagram with various elements and connectors.

Overlaid on these screenshots are logos for Visual Studio Code (blue), Chrome (red, yellow, green, blue), GitHub (black), and Red Hat (red and white).



The image shows a VS Code window with two Envelope Instance editors side-by-side. The left editor is titled 'a.dmn' and contains a diagram with a single element labeled 'BusinessKnowl edgeModel-2'. The right editor is titled 'eder.bpmn' and contains a diagram with a single element labeled 'Task'. Both editors are highlighted with a blue border. The VS Code interface includes an Explorer sidebar on the left showing a project structure with files 'a.dmn', 'eder.bpmn', 'order.bpmn2', and 'test.bpmn'. The top of the window shows the file names 'a.dmn' and 'eder.bpmn'. The text 'VS Code Channel' is in the top left, and 'VS Code Channel' is in the bottom left. The text 'DMN Editor' and 'BPMN Editor' are at the bottom of their respective editors. The text 'Envelope Instance' is at the bottom of each editor's frame.

VS Code Channel

DMN Editor

BPMN Editor

Envelope Instance

Envelope Instance

github.com/ederign/demo/blob/master/order.bpmn2

Search or jump to... Pull requests Issues Marketplace Explore

ederign / demo Unwatch 1 Star 0 Fork 1

<> Code Issues Pull requests 1 Actions Projects Wiki Security Insights Settings

master demo / order.bpmn2 Go to file ...

ederign adding some samples Latest commit e2b4bea 2 days ago History

1 contributor

This is a readonly visualization See as source Open in Online Editor Copy link to Online Editor Full Screen

754 lines (754 sloc) 50.3 KB Raw Blame

```
graph LR; subgraph Restaurant; Start(( )) --> Accept[Accept Order]; end; Accept --> Accepted{Accepted?}; Accepted -- Yes --> Ready[Order Ready]; Accepted -- No --> Accept; Ready --> End(( )); subgraph Restaurant; Ready --- Register[Restaurant: Register that the order is ready]; end; subgraph Restaurant; CallCenter[Restaurant: Call Center]; end;
```

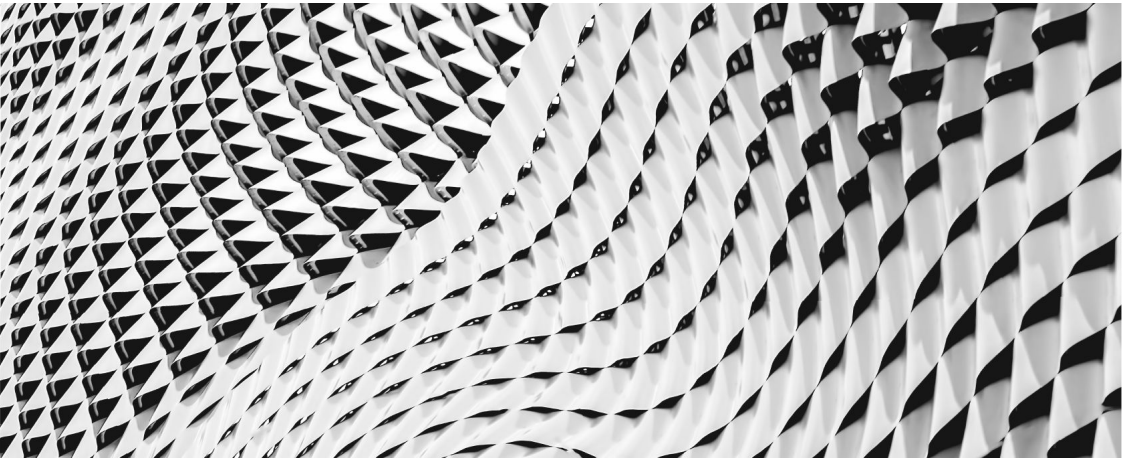
BPMN Editor

Envelope Instance

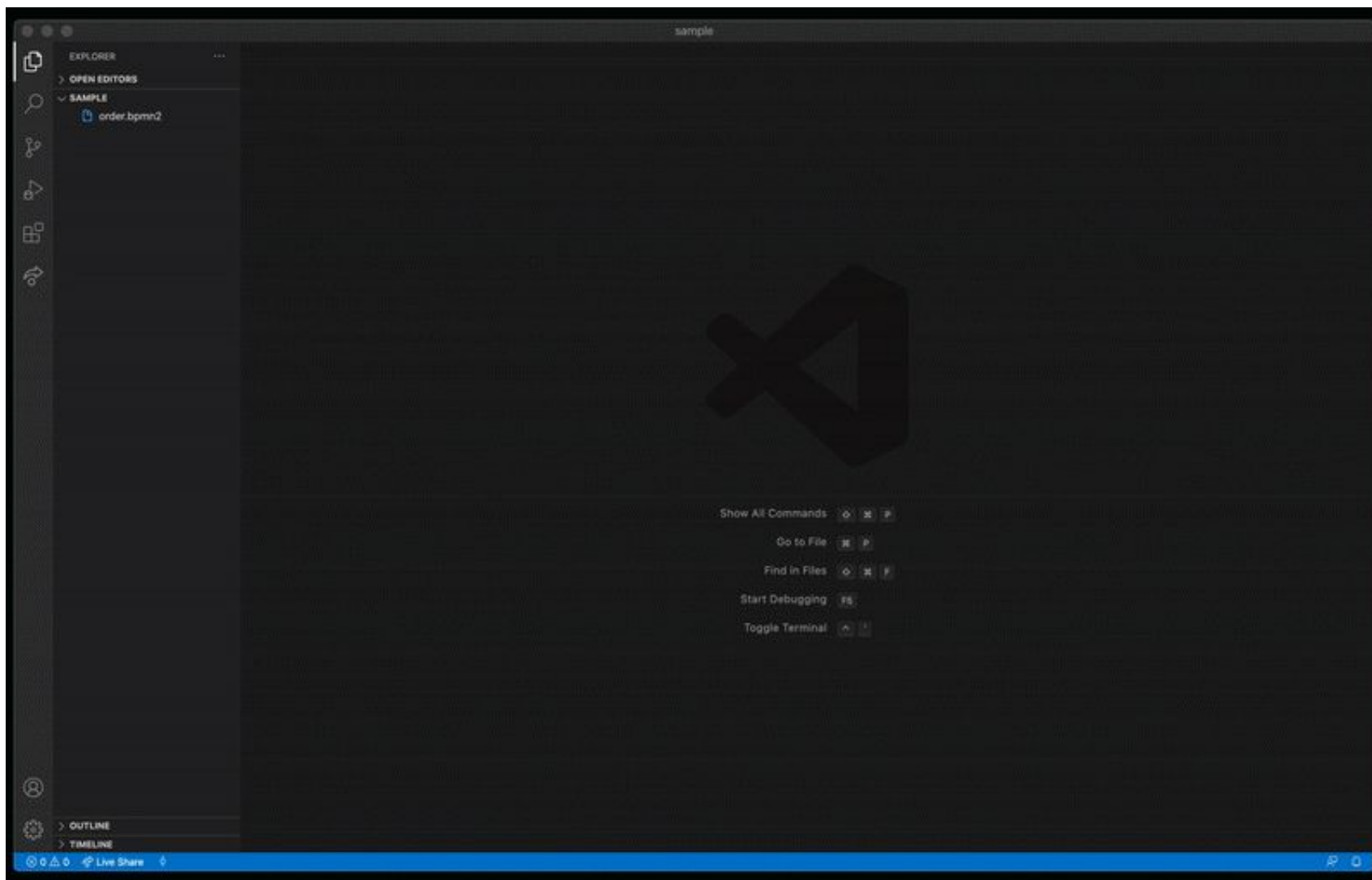
Chrome Extension Channel

© 2020 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

Feature Highlights



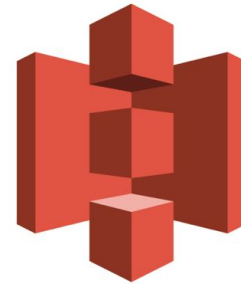
The Multiplying Architecture



VSCode Native



Google
Drive

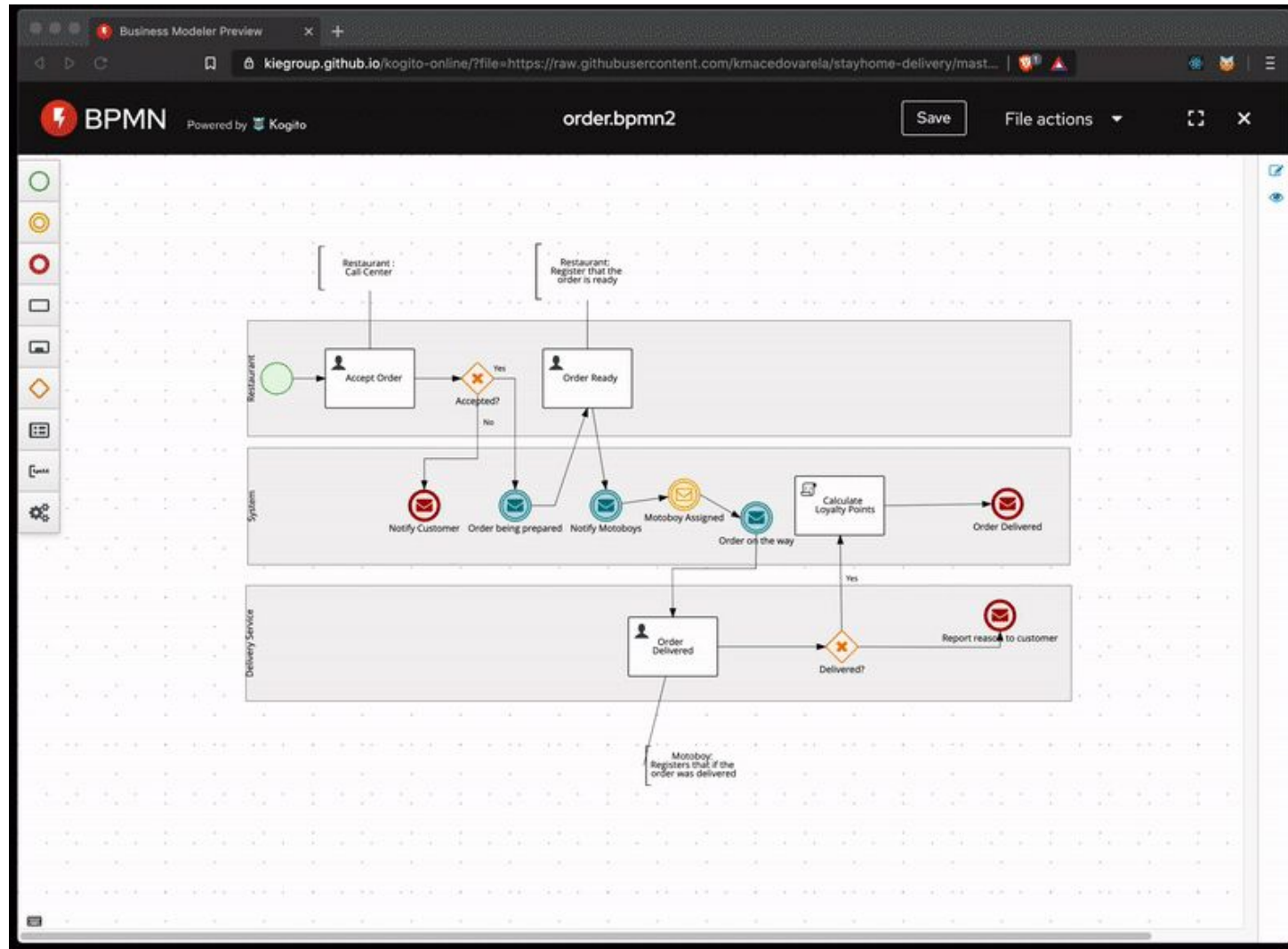


Amazon S3



Unified I/O API

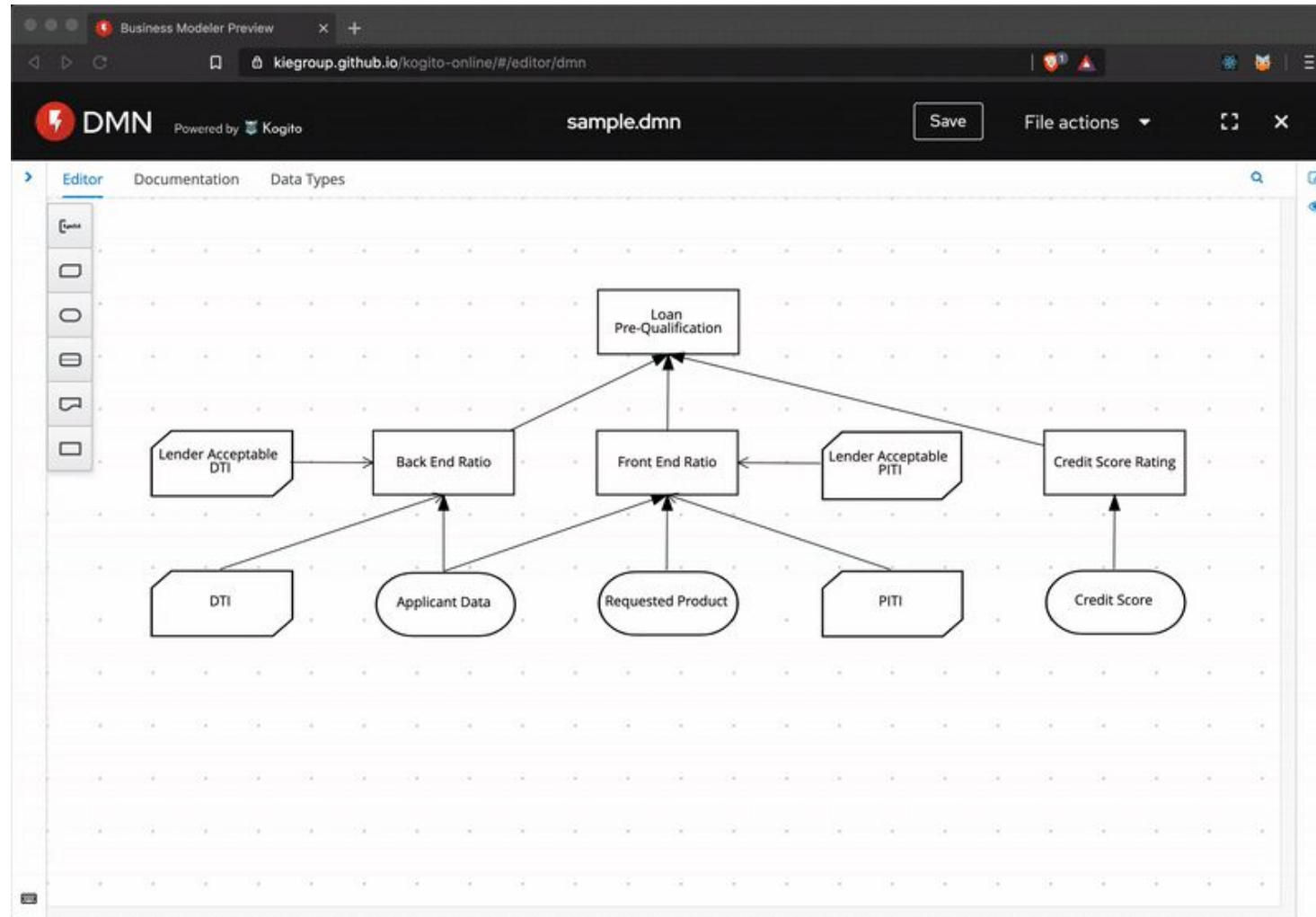
Read and Write content
from multiple sources like
GitHub, Gist, FileSystem,
S3 (soon), etc.



State Control

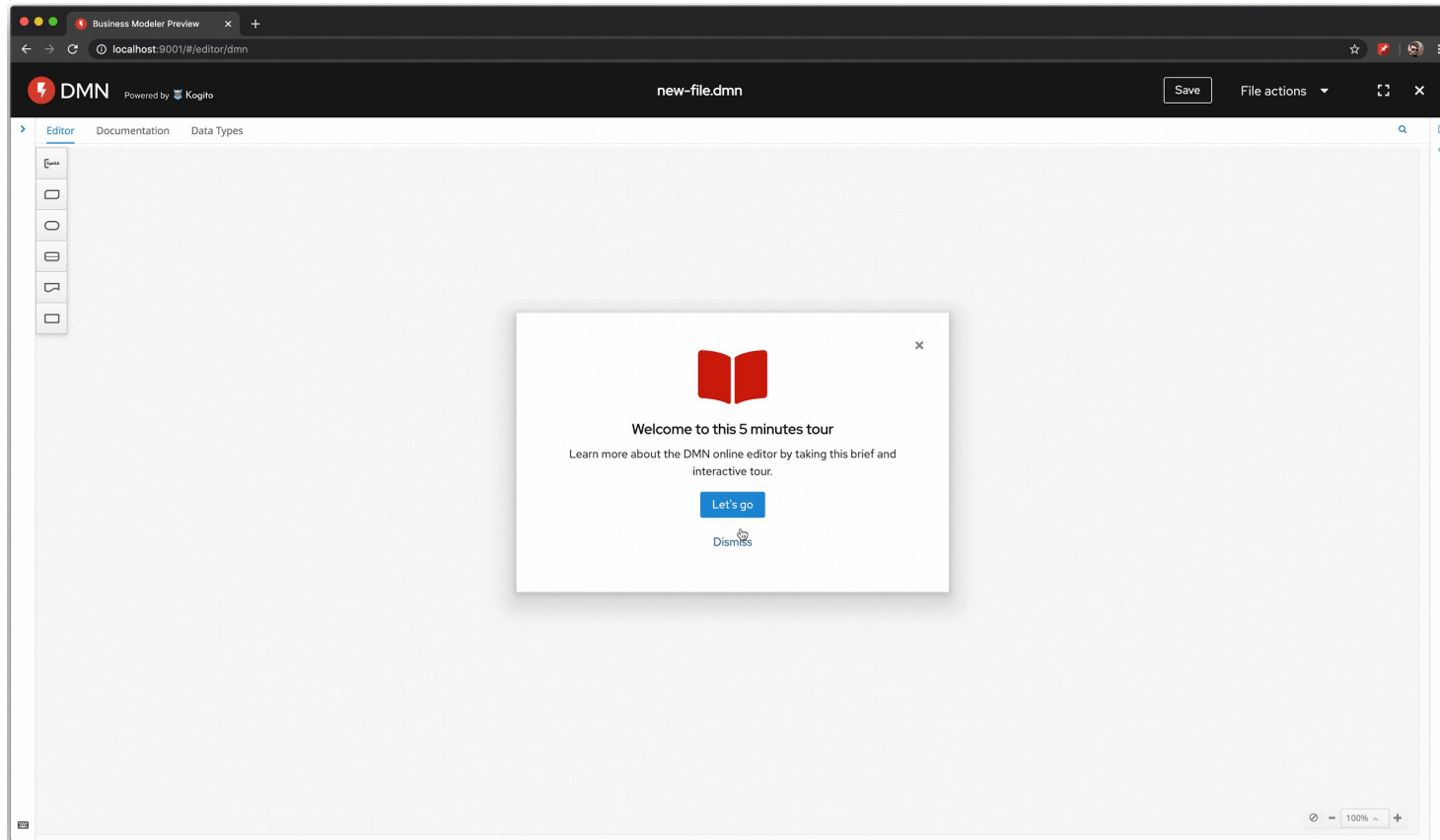
Cross channel support for Undo, Redo, and Dirty detection.

The Multiplying Architecture



Keyboard Shortcuts

The Multiplying Architecture



Guided Tour

The Multiplying Architecture

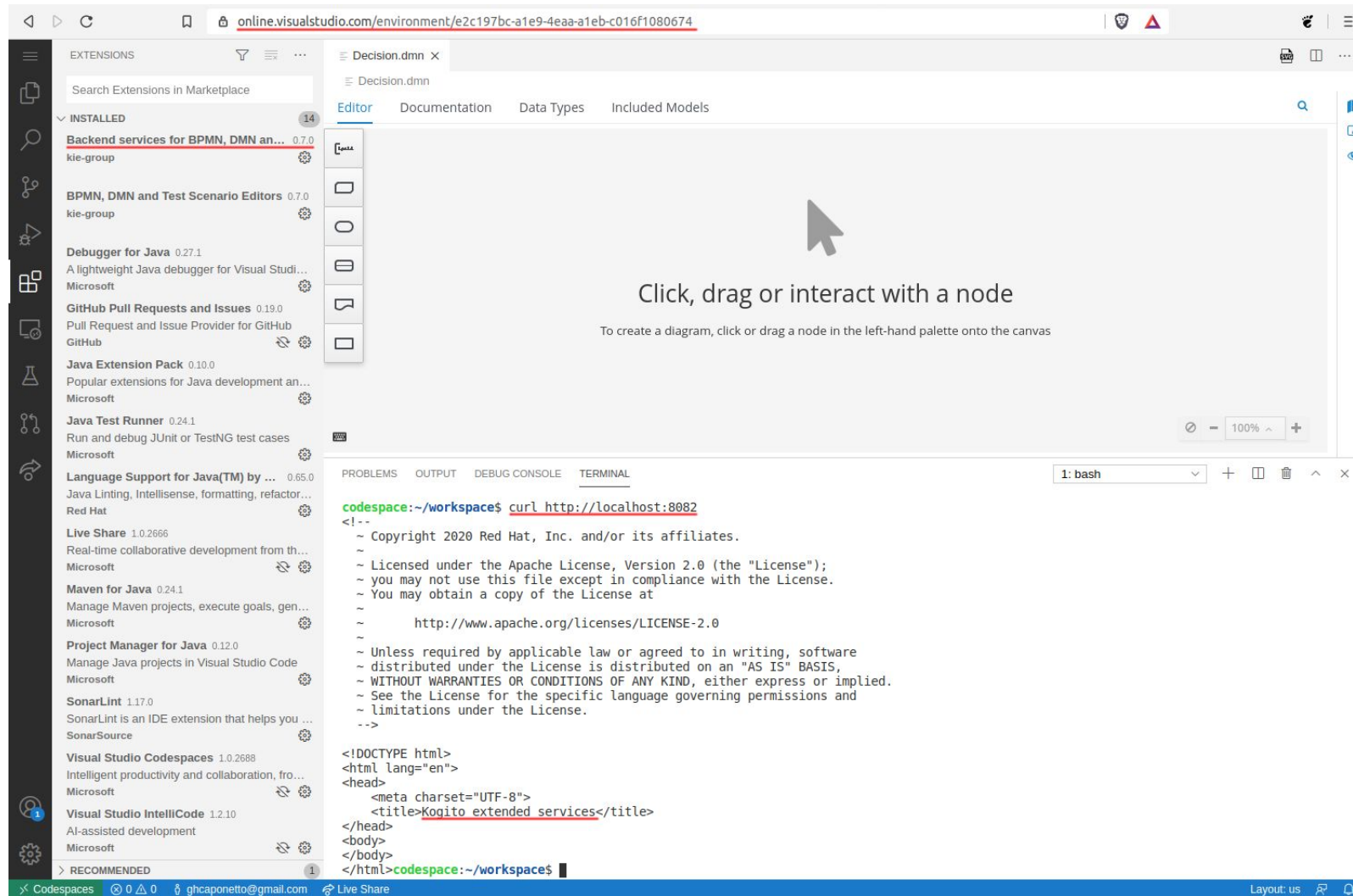
The image shows a BPMN editor interface with a process flow diagram and a GitHub Gist containing the BPMN XML code. The process flow diagram includes a start event 'travelers', a task 'Process Traveler', a decision diamond 'Processed Traveler?', a task 'Log Traveler', and an end event 'processedtraveler'. The GitHub Gist shows the XML code for the BPMN process, including definitions for items, messages, and the process flow itself.

```
1 <bpmn2:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:bpmn2="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:bp
2 <bpmn2:itemDefinition id="_travellerItem" structureRef="org.acme.travel.Traveller"/>
3 <bpmn2:itemDefinition id="__0E8784C3-1BEC-4A51-A5E6-D7E5D3A4482_eventInputXitem" structureRef="org.acme.travel.Traveller"/>
4 <bpmn2:itemDefinition id="processedtravellersType" structureRef="org.acme.travel.Traveller"/>
5 <bpmn2:itemDefinition id="__60FA6326-76DC-4DB2-AB86-DB8AC8EE8DC8_namespaceInputXitem" structureRef="java.lang.String"/>
6 <bpmn2:itemDefinition id="__60FA6326-76DC-4DB2-AB86-DB8AC8EE8DC8_modelInputXitem" structureRef="java.lang.String"/>
7 <bpmn2:itemDefinition id="__60FA6326-76DC-4DB2-AB86-DB8AC8EE8DC8_decisionInputXitem" structureRef="java.lang.String"/>
8 <bpmn2:itemDefinition id="__60FA6326-76DC-4DB2-AB86-DB8AC8EE8DC8_travellerInputXitem" structureRef="org.acme.travel.Traveller"/>
9 <bpmn2:itemDefinition id="__60FA6326-76DC-4DB2-AB86-DB8AC8EE8DC8_travellerOutputXitem" structureRef="org.acme.travel.Traveller"/>
10 <bpmn2:itemDefinition id="__8BEA9396-93DE-4D44-8CE2-4A146464264E_eventOutputXitem" structureRef="org.acme.travel.Traveller"/>
11 <bpmn2:itemDefinition id="travellersType" structureRef="org.acme.travel.Traveller"/>
12 <bpmn2:message id="_2620EamLED167ZUJF8Z28A" itemRef="processedtravellersType" name="processedtravellers"/>
13 <bpmn2:message id="_2620EamLED167ZUJF8Z28A" itemRef="travellersType" name="travellers"/>
14 <bpmn2:process id="travelers" drools:packageName="org.kie.kogito.test" drools:version="1.0" drools:adhoc="false" name="Process Travelers"
15 <bpmn2:property id="traveller" itemSubjectRef="_travellerItem" name="traveller"/>
16 <bpmn2:sequenceFlow id="_D15CD483-D31D-42F8-A93A-AAAF44292D84" sourceRef="__8BEA9396-93DE-4D44-8CE2-4A146464264E" targetRef="__60FA6326-7
17
18 <drools:metaData name="isAutoConnection.source">
19 <drools:metaValue><![CDATA[true]]></drools:metaValue>
20 </drools:metaData>
21 <drools:metaData name="isAutoConnection.target">
22 <drools:metaValue><![CDATA[true]]></drools:metaValue>
23 </drools:metaData>
24 </bpmn2:extensionElements>
25 </bpmn2:sequenceFlow>
```

Gist as storage

Pure client side mechanism
to store content using
GitHub Gists.

The Multiplying Architecture



The screenshot shows the Visual Studio Code interface. On the left is the Extensions Marketplace with a list of installed extensions including 'Backend services for BPMN, DMN an...', 'BPMN, DMN and Test Scenario Editors', 'Debugger for Java', 'GitHub Pull Requests and Issues', 'Java Extension Pack', 'Java Test Runner', 'Language Support for Java(TM) by ...', 'Live Share', 'Maven for Java', 'Project Manager for Java', 'SonarLint', 'Visual Studio Codespaces', and 'Visual Studio IntelliCode'. The main editor area displays a canvas with a mouse cursor and the text 'Click, drag or interact with a node'. Below the editor is a terminal window showing the command `curl http://localhost:8082` and its output, which includes a license notice and HTML content.

```
codespace:~/workspace$ curl http://localhost:8082
<!--
~ Copyright 2020 Red Hat, Inc. and/or its affiliates.
~ Licensed under the Apache License, Version 2.0 (the "License");
~ you may not use this file except in compliance with the License.
~ You may obtain a copy of the License at
~
~     http://www.apache.org/licenses/LICENSE-2.0
~
~ Unless required by applicable law or agreed to in writing, software
~ distributed under the License is distributed on an "AS IS" BASIS,
~ WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
~ See the License for the specific language governing permissions and
~ limitations under the License.
-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Kogito extended services</title>
</head>
<body>
</body>
</html>codespace:~/workspace$
```

Backend Services

A pluggable infrastructure, able to augment the capabilities of the views and editors by enabling some backend dependent features.

Embedded Editors (soon)

```
const dmnEditor = kieTooling.dmnEditor('targetEmbeddedDiv', {standalone:true})
```

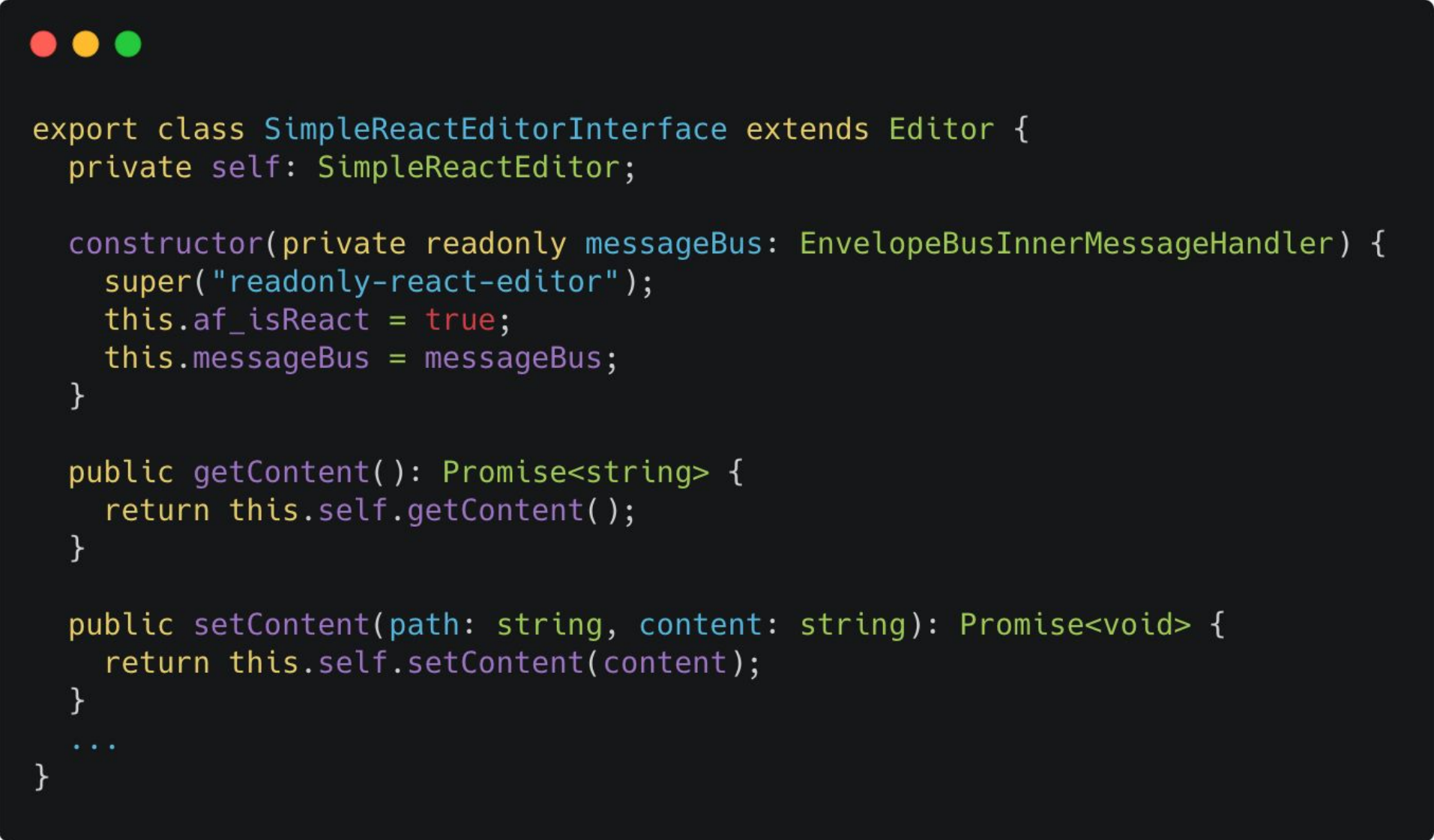
```
const bpmnEditor = kieTooling.bpmnEditor('targetEmbeddedDiv', {standalone:true})
```

Multiplying Architecture In Detail

<https://github.com/kiegroup/kogito-tooling-examples>



```
/**
 * Editor component API. Basic Editor feature definitions.
 */
export interface EditorApi {
  setContent(path: string, content: string): Promise<void>;
  getContent(): Promise<string>;
  getPreview(): Promise<string | undefined>;
  getElementPosition(selector: string): Promise<Rect | undefined>;
  undo(): Promise<void>;
  redo(): Promise<void>;
}
```



```
export class SimpleReactEditorInterface extends Editor {
  private self: SimpleReactEditor;

  constructor(private readonly messageBus: EnvelopeBusInnerMessageHandler) {
    super("readonly-react-editor");
    this.af_isReact = true;
    this.messageBus = messageBus;
  }

  public getContent(): Promise<string> {
    return this.self.getContent();
  }

  public setContent(path: string, content: string): Promise<void> {
    return this.self.setContent(content);
  }
  ...
}
```


```
export class SimpleReactEditor extends React.Component<Props, State> {
  constructor(props: Props) {
    super(props);
    props.exposing(this);
    this.state = {
      content: ""
    };
  }
  ...

  public async setContent(content: string): Promise<void> {
    this.setState({ content: content });
  }

  public async getContent(): Promise<string> {
    return this.state.content;
  }

  public render() {
    return (
      <textarea
        style={{
          width: "100%",
          height: "100%",
          outline: 0,
          boxSizing: "border-box",
          border: 0,
          color: "black"
        }}
        value={this.state.content}
        onChange={(e: any) => this.updateContent(e.target.value)}
      />
    );
  }
}
```

```
export class SimpleReactEditorsRoutes implements Routes {
  public getRoutes() {
    return new Map<string, SimpleReactEditorsLanguageData>([
      [
        "txt",
        {
          type: "my-editor-type",
          anyData: "something"
        }
      ]
    ]);
  }
}
```



```
export function activate(context: vscode.ExtensionContext) {
  console.info("Extension is alive.");

  KogitoVsCode.startExtension({
    extensionName: "kogito-tooling-examples.vscode-extension-pack-simple-react",
    webviewLocation: "dist/webview/index.js",
    context: context,
    viewType: "kieKogitoWebviewSimpleEditors",
    getPreviewCommandId: "",
    router: new DefaultVsCodeRouter(context, new SimpleReactEditorsRoutes())
  });

  console.info("Extension is successfully setup.");
}
```

EXPLORER

- OPEN EDITORS
- KOGITO-TOOLING-EXAMPLES
 - node_modules
 - packages
 - chrome-extension-pack-simple-react
 - simple-react-editors
 - vscode-extension-pack-simple-react
 - .vscode
 - dist
 - extension
 - fonts
 - images
 - webview
 - kogito_tooling_examples_vscode_extensi...
 - node_modules
 - src
 - extension
 - extension.ts
 - webview
 - .gitignore
 - .vscodeignore
 - LICENSE
 - package.json
 - README.md
 - tsconfig.json
 - webpack.config.js
 - .gitignore
 - lerna.json
 - LICENSE
 - package.json
 - prettier.config.js
 - README.md
 - test.txt
 - tsconfig.json
 - tslint.json
 - update_version_to.js
 - yarn.lock

OUTLINE

TIMELINE

NPM SCRIPTS



- Show All Commands
- Go to File
- Find in Files
- Start Debugging
- Toggle Terminal



```
import { startExtension, DefaultChromeRouter } from "@kogito-tooling/chrome-extension";
import { SimpleReactEditorsRoutes } from "simple-react-editors";

startExtension({
  name: "KIE :: Kogito Simple React Editor",
  editorIndexPath: "envelope/index.html",
  extensionIconUrl: chrome.extension.getURL("/resources/kie-icon.png"),
  githubAuthTokenCookieName: "github-oauth-token-kie-editors",
  router: new DefaultChromeRouter(new SimpleReactEditorsRoutes())
});
```


The screenshot shows a GitHub repository page for 'ederign/demo'. The browser address bar shows 'github.com/ederign/demo/'. The repository name 'ederign / demo' is at the top left. Navigation tabs include 'Code', 'Issues', 'Pull requests' (with a count of 1), 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. On the right, there are buttons for 'Unwatch' (1), 'Star' (0), and 'Fork' (1). Below the repository name, there are buttons for 'Go to file', 'Add file', and 'Code'. A commit history table is displayed, showing the most recent commit by 'ederign' updating 'test.txt' 13 seconds ago. Below the table, the 'README.md' content is shown, containing the word 'demo'. On the right side, there are sections for 'About' (no description), 'Releases' (no releases published), and 'Packages' (no packages published). The footer contains copyright information for GitHub, Inc. and various links like 'Terms', 'Privacy', 'Security', 'Status', 'Help', 'Contact GitHub', 'Pricing', 'API', 'Training', 'Blog', and 'About'.

github.com/ederign/demo/

Search or jump to... Pull requests Issues Marketplace Explore Create token Place your token here... Reset

ederign / demo Unwatch 1 Star 0 Fork 1

<> Code Issues Pull requests 1 Actions Projects Wiki Security Insights Settings

master - 1 branch 0 tags Go to file Add file Code

ederign Update test.txt 3efb542 13 seconds ago 8 commits

LICENSE	Initial commit	8 days ago
README.md	Initial commit	8 days ago
order.bpmn2	adding some samples	8 days ago
sample.bpmn	adding some samples	8 days ago
sample.dmn	-	8 days ago
test.txt	Update test.txt	13 seconds ago

README.md

demo

About No description, website, or topics provided. Readme Apache-2.0 License

Releases No releases published Create a new release

Packages No packages published Publish your first package

© 2020 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

Why do we need a **new** architecture?

Goals of The Multiplying Architecture

solve a problem



Multiple Distributions

The origin of multiplying architecture is rooted in the need to distribute the same set of components in a myriad of platforms.



Minimize code changes

The components to be distributed should be preserved untouched and with avoiding feature flags.



Bridge

It has to embrace different generations of technology stack.



Questions


Thank you

Eder Ignatowicz

Principal Software Engineer

@ederign

 linkedin.com/company/red-hat

 youtube.com/user/RedHatVideos

 facebook.com/redhatinc

 twitter.com/RedHat