



VIRTUAL Developer Bench

2021

A laptop screen displaying a virtual meeting with multiple participants. The participants are shown in a grid layout, with a larger central window showing a man with a beard and glasses. The laptop is on a wooden desk, and a hand is visible at the bottom, typing on the keyboard.

Sabre Red 360 SDK: Understanding Feature Fundamentals

The Developer Experience Team

1 December 2021

On the Call



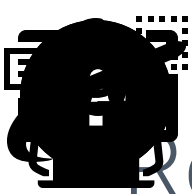
Alexandre Meneghello
Software Engineering



Avery Perkins
Product Marketing

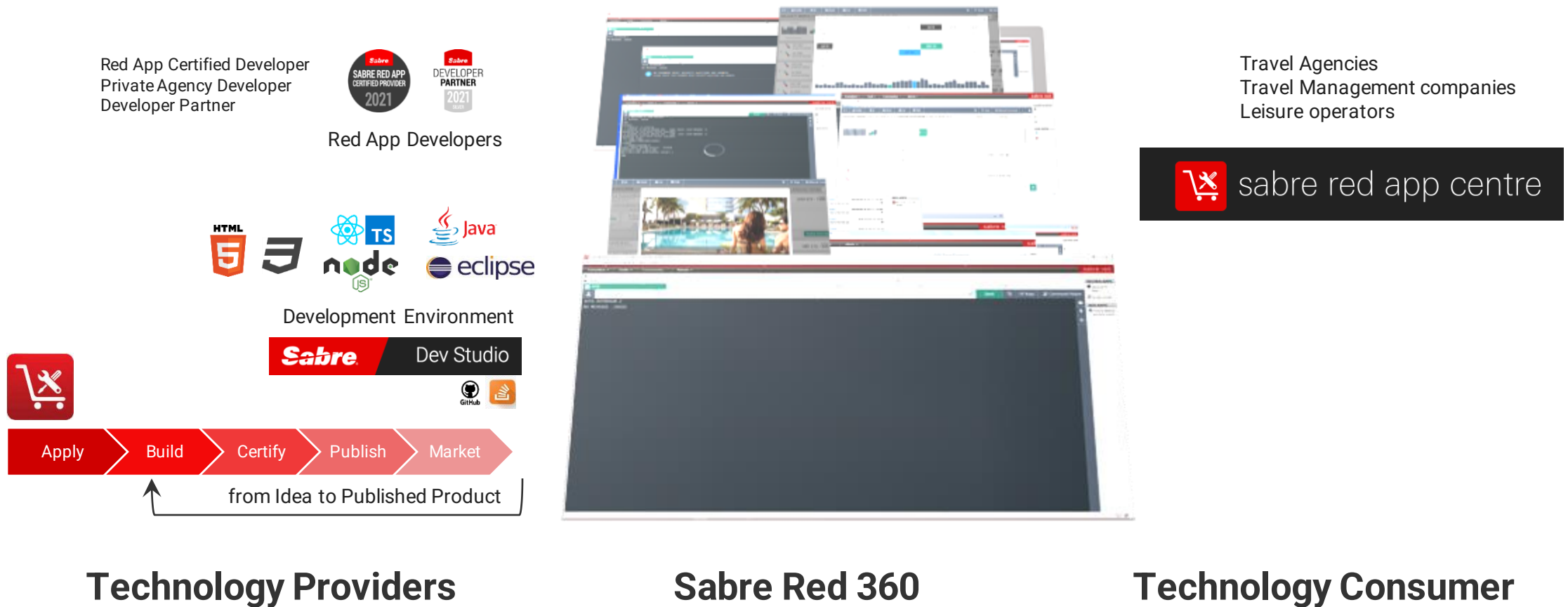


Alex Xavier
Product Marketing

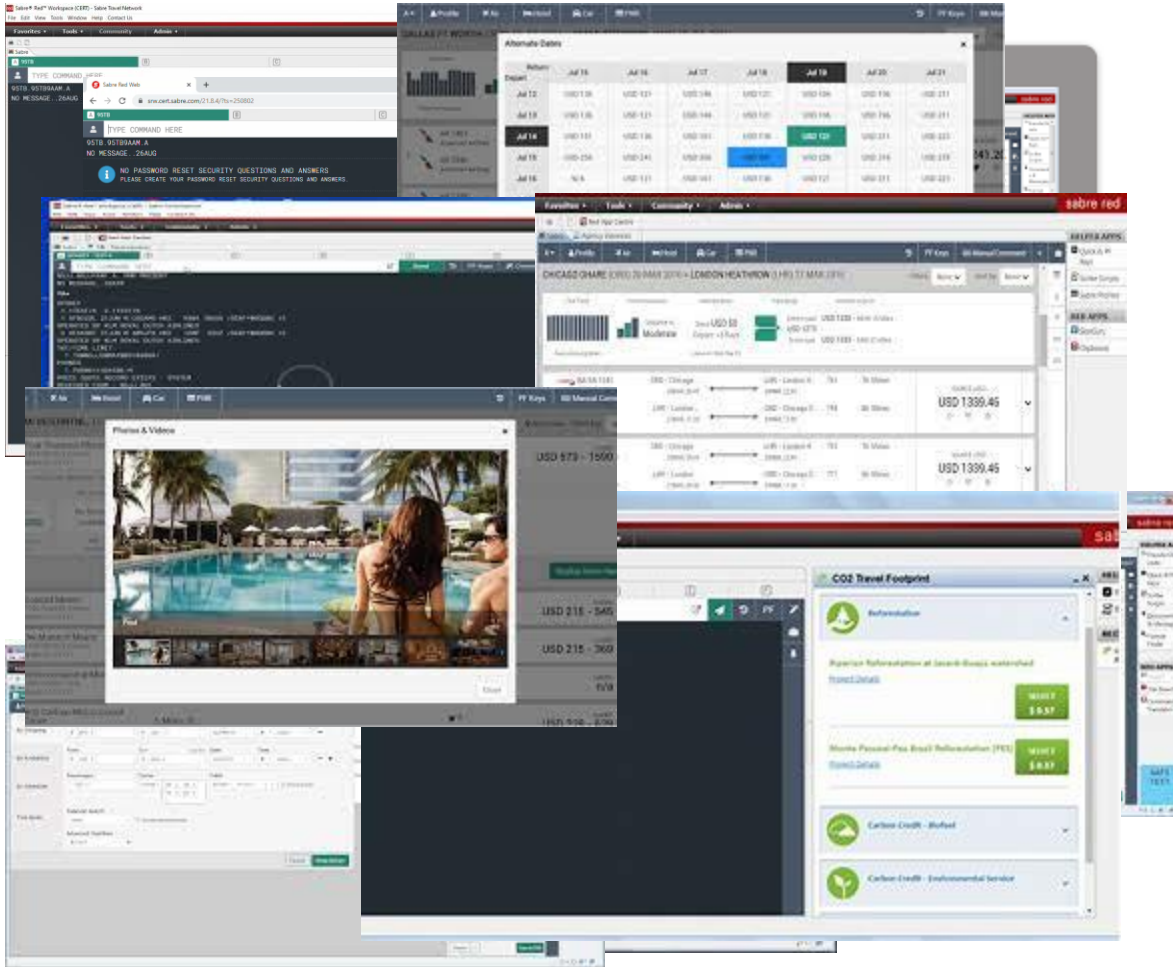


Red App ecosystem overview

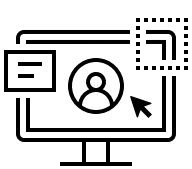
Sabre Red 360 is responsible for an entire ecosystem involving Travel technology producers and consumers, through its innovative Red App Marketplace and Developer network.



Sabre Red 360 & customization



- Sabre Red 360 is an end-to-end sales workspace that sells billions in Travel inventory each year. Could be extended by plugging-in modules, or Red Apps.
- Sabre Red 360 main View, offers advanced visualization capabilities and orchestration to the Travel consultant workflow.
 - Workflow, from Sabre Red 360 core, any business operation involved on **SHOP, SELL, BOOK, TICKET** and **MANAGE RESERVATIONS** of FLIGHT, HOTEL, CAR, CRUISE AND RAIL
- Sabre Red 360 Desktop App, it is a multi-windowed workspace application, which hosts Sabre's main view, complimentary user workflows and allows integrations with back-office and external systems.
- Sabre Red Web, Mobile design brings (big) part of sales workspace to Travel Consultant's pocket, or an internet café anywhere.

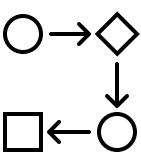


Contribute

Create Sabre Red native experience, create custom forms and widgets to communicate effectively with users.

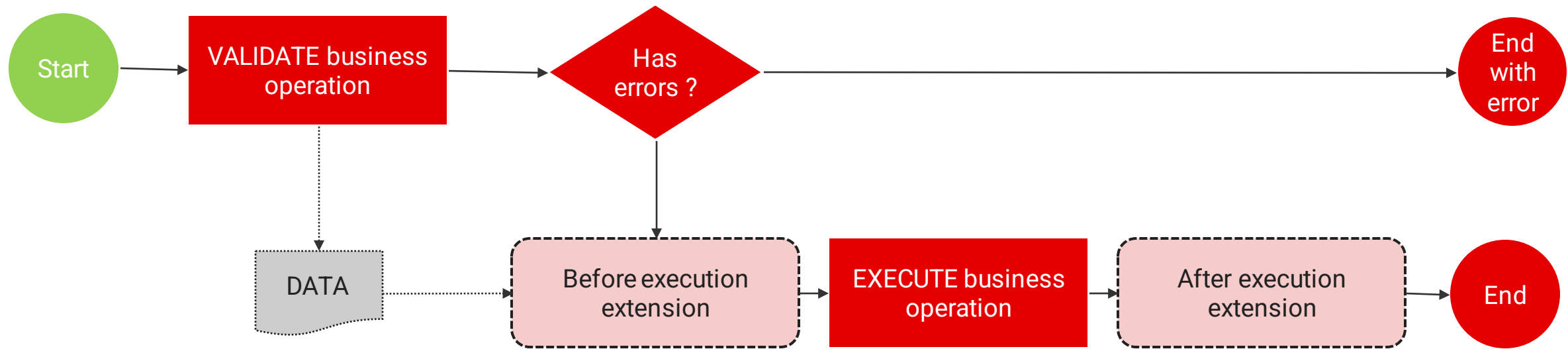
The collage displays various components of the Sabre Red user interface:

- Search Forms:** Includes a 'Search Profile' dialog with fields for 'Search in' (Current PCC), 'Search Type' (Traveler Name), 'Last Name', and 'Given Name'. It also features a 'Shop Hotels' button and an 'Add to PNR' option.
- Flight Results:** Shows a search for 'Air Shopping DFW - Dallas → MIA - Miami' on 'Mon, 16 Jul - Fri, 20 Jul'. It includes filters for 'FARE TREND' (Fares are going up), 'TRAVEL SEASONALITY' (Moderate), 'FLEXIBLE DATES' (Similar prices ± 3 days), 'FARE RANGE' (USD 365), and 'ALTERNATE AIRPORTS' (USD 423 DAL). A red-bordered box highlights a 'Packages available!' notification with an 'Enter website' button.
- Flight Details:** Displays a flight from 'JFK - New York → LAX - Los Angeles' on 'Mon, 25 Sep - Fri, 29 Sep'. It shows fare trends, travel seasonality, and a table of flight options with prices like '+USD 323.60' and '+USD 323.60'.
- Command Window:** A 'TYPE COMMAND HERE' input field with a 'Send' button and a 'Command Helper' link. Below it, a 'Custom Workflow' section lists actions like 'New PNR', 'Modify PNR', and 'Order PNR'.
- Modal Header Title:** A dialog box with the title 'Modal Header Title' and a body text area for user assistance.
- Flight Summary:** A detailed view of a flight from 'DFW - Dallas' to 'MIA - Miami' on '06NOV, 16:50' to '06NOV, 20:33'. It includes flight details like 'AA 905', 'AA 207', 'American Airlines', and a total price of 'USD 196.40'.

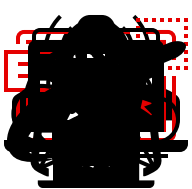


Automate

Customize pre-defined workflows, reduce or augment standard workflows seamlessly

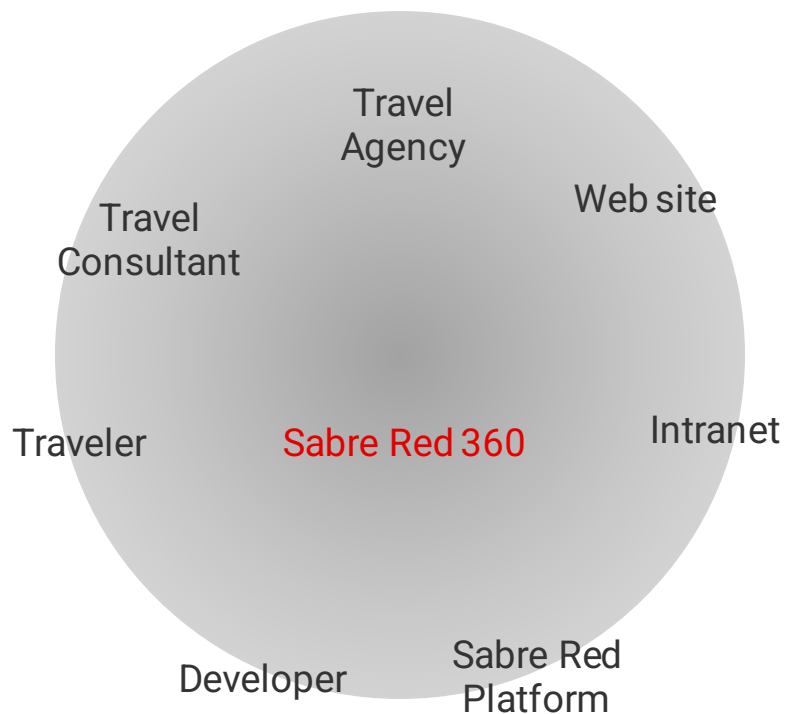


SHOP, PRICE	BOOK	SELL	TICKET	MANAGE
Air, Hotel Exchange Hotel Details	Air, Hotel	Air, Hotel, Car Passive Add	Air	End PNR Modify PNR Queue Place



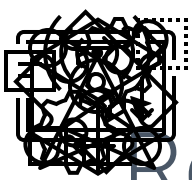
Integrate

Integrate adjacent systems, direct access to external workflows, content sources and APIs



Integration opportunities:

- ✓ Connectivity with external systems
- ✓ Back-office integration
- ✓ API client
- ✓ Database integration
- ✓ Customer profiles
- ✓ Intranet / Extranet
- ✓ integrate existing web sites to the SR360 workspace



Red App SDK, fundamental building blocks

Red App is a software package, created with Sabre Developer Tools, which bundles different extensions to customize Sabre Red 360 operations during runtime.

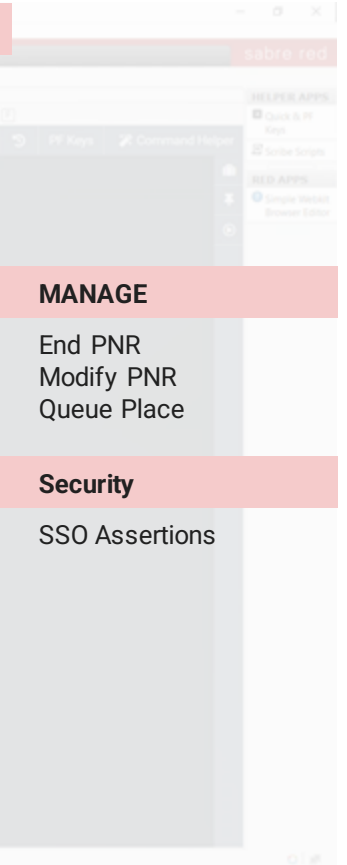
Contributions to the User Interface, allows Red Apps to customize the Travel consultant workspace (Eclipse) or the primary Sabre Red 360 UI (NGV)

Workflow extensions, allow access to data and controlled execution of Business operations done through Sabre Red 360 main View

Communication Foundations, allows Red Apps to access Sabre Red Platform Services exposed through SR 360 runtime and abroad

Tools and Utilities, general utility layer, allows to deal with user settings and authentication, as well integration with external systems and process automation.

UI Widgets	Workflow	Web Browser	Workspace	
Pop-up Forms Tile Widgets Command button + Popover React bootstrap	Workflow panel button Nudge Add to PNR form Refresh PNR Interstitial	CrossWindow WebKit View / Editor Javascript API	View / Editor window Red App Sidebar Workspace Menus Notifications Modal Dialog	
SHOP, PRICE	BOOK	SELL	TICKET	MANAGE
Air, Hotel Exchange Hotel Details	Air, Hotel	Air, Hotel, Car Passive Add	Air	End PNR Modify PNR Queue Place
Sabre APIs	Sabre Commands	Web	Active Listening	Security
REST and SOAP API calls GetReservation	Execute Sabre Commands LLS Calls - HOST	HTTP Request	EMU_RESPONSE Event EMU_COMMAND Service	SSO Assertions
Agency	Support	Workspace		
Agent Profile Config Services	Log services First run	Plugin resources Internationalization		





RED APP EXAMPLE



Let's create a Red App to demonstrate common use cases and building blocks provided by the Red App SDK.

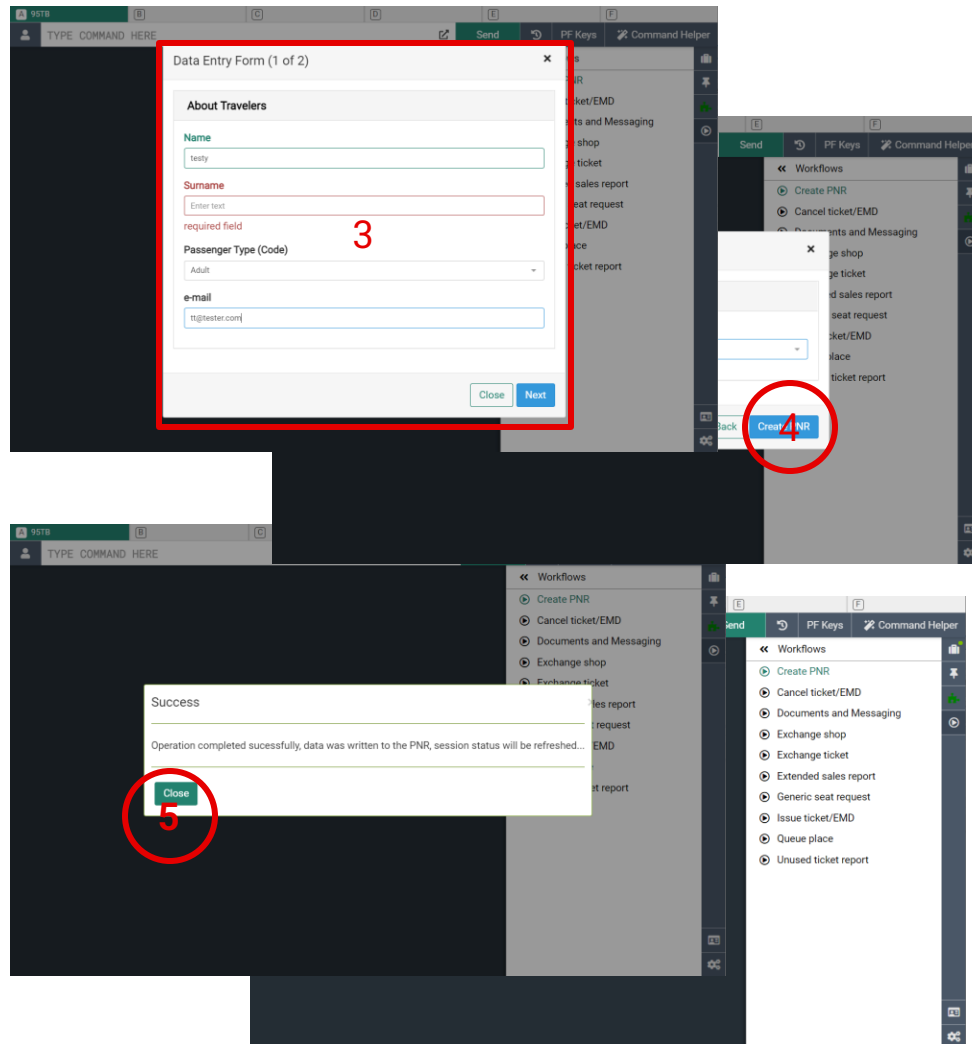
Red App project – PNR Shell

- The main use case for this Red App would be to prepare a new Reservation record, gathering details about the trip and travelers it will help personalize content during travel consultant workflow.
- User will start workflow by clicking Create PNR button on the Sidebar panel.
- A multi-stage form will be presented to capture information about passenger and travel details.
- Once submitted, the information gathered will be written to the current PNR on the session.

The image displays a sequence of four overlapping screenshots of the Red App interface, illustrating the workflow for creating a PNR (Passenger Name Record).

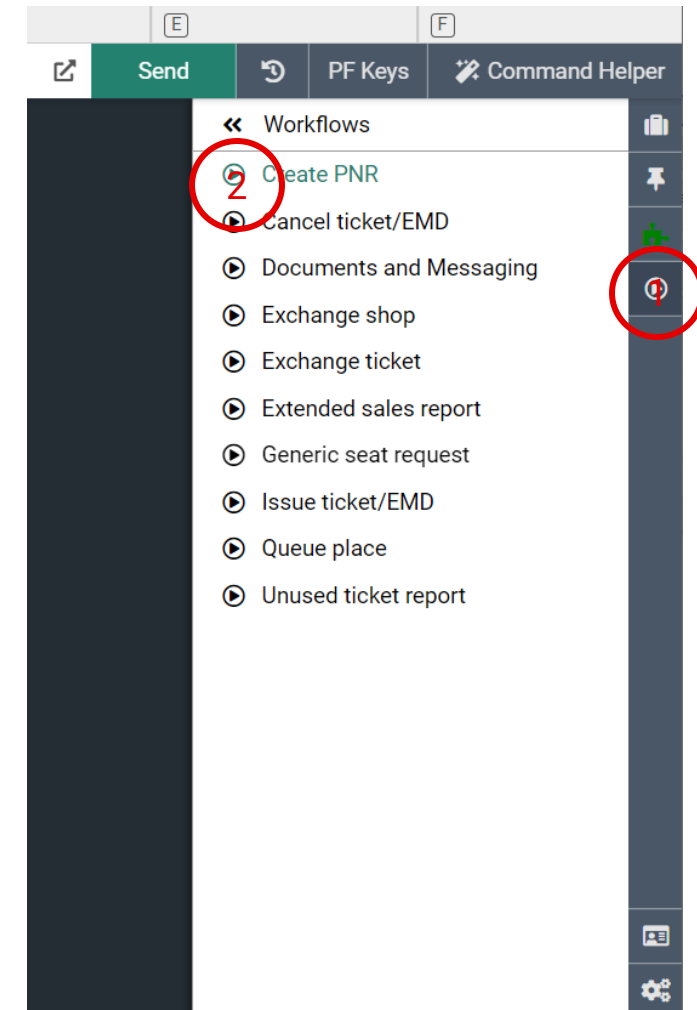
- Top Screenshot:** Shows the main sidebar with the 'Create PNR' button highlighted. The sidebar also includes 'Cancel ticket/EMD', 'Documents and Messaging', and 'Exchange shop'.
- Second Screenshot:** Shows the 'Data Entry Form (1 of 2)' with the 'About Travelers' section. The form includes fields for Name, Surname, Passenger (Adult), and email (td@tester.co).
- Third Screenshot:** Shows the 'Data Entry Form (2 of 2)' with the 'Travel' section. The form includes fields for Travel (Jesum) and a 'Save' button.
- Bottom Screenshot:** Shows a 'Success' message: 'Operation completed successfully, data was written to the PNR, session status will be refreshed...'. A 'Close' button is visible.

Implementation details



SDK features used :

1. Contribution to custom workflow panel
2. Sidebar button
3. Custom pop-up Dialog provided by **Layer Service** and react-bootstrap component library
4. Upon form submission Red App utilizes **SoapAPI Service** and send an XML request to **UpdateReservationRQ** API
5. After writing data to PNR, Red App user **PnrPublic Service** to refresh the front-end session



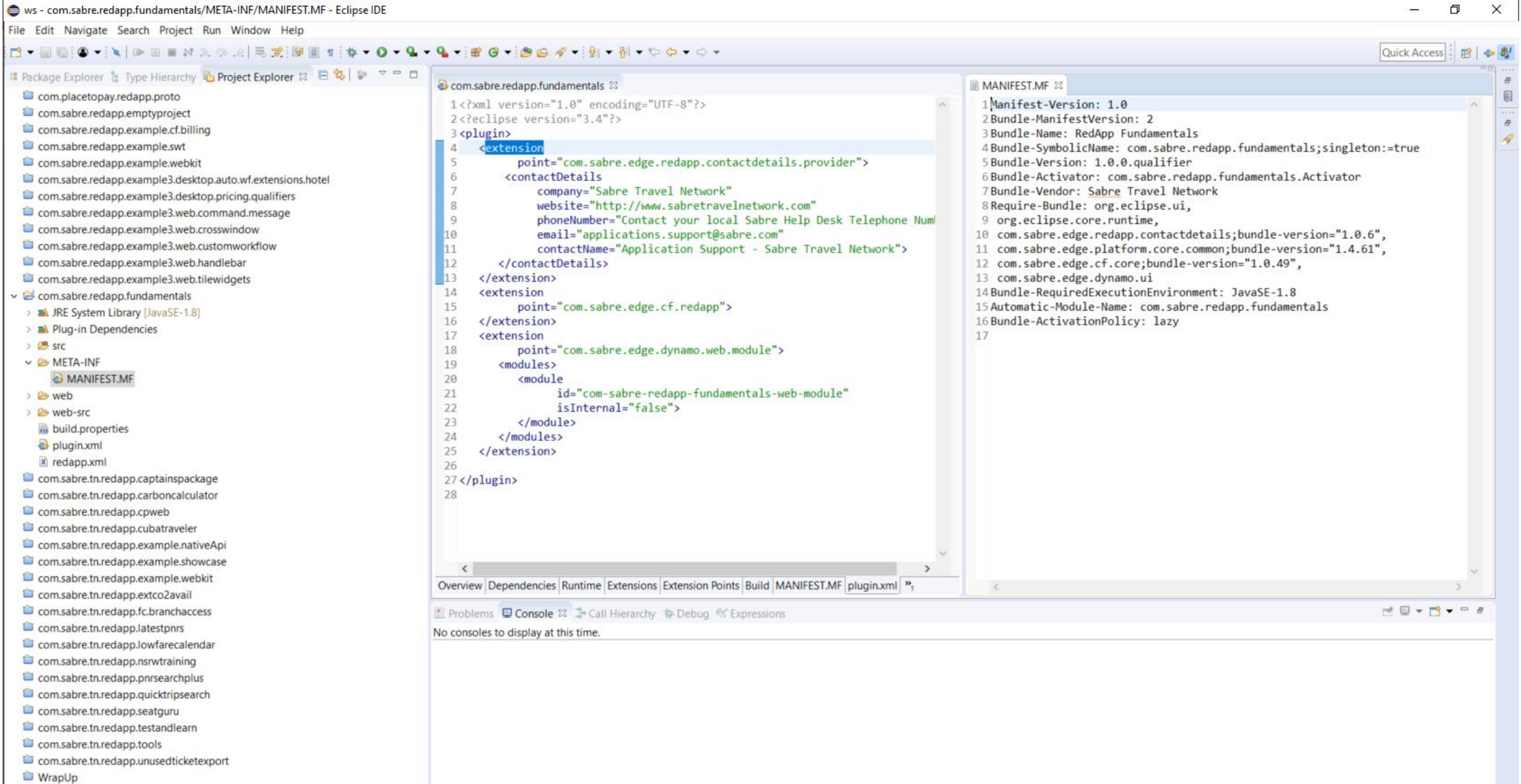


DEVELOPMENT SLIDESHOW

Walk through implementation of a Red App that presents a form to the user, collect some data and consume Sabre API service to write information to the reservation. (PNR)

*** The source code for this project could be found at:

[GitHub - SabreDevStudio/RedApp-Fundamentals: Sabre sample Web Red App showing SDK fundamentals](https://github.com/SabreDevStudio/RedApp-Fundamentals)



We start with a Red App project structure under Eclipse IDE, so we can have both developer platforms available



Q

99



11

We



Im

Web Red App start with a Module implementation on Main.ts file, there's where the plug-in lifetime is implemented, during init() method we add sidebar menu configuration

```

TS Main.ts TS CreatePNR.tsx X
src > code > TS CreatePNR.tsx > ...
1  import * as React from "react";
2  import {Button, Form, FormGroup, Modal, InputGroup, ControlLabel, FormControl, HelpBlock, Panel, Alert} from "react-bootstrap";
3  import { getService, t } from "../Context";
4  import { LayerService } from "sabre-ngv-core/services/LayerService";
5  import { ISoapApiService } from "sabre-ngv-communication/interfaces/ISoapApiService";
6  import { PnrPublicService } from "sabre-ngv-app/app/services/impl/PnrPublicService";
7
8  /*
9  Define interface for handling Traveler details on the React component state
10 */
11 export interface Traveler {
12     name:string,
13     surname:string,
14     email?:string,
15     typeCode?:'ADT' | 'INF' | 'CNN',
16 }
17
18 /*
19 Define interface for handling field validation on the React component state
20 */
21 export interface FieldValidation {
22     [fieldId:string]:{
23         isValid:boolean,
24         status:"error"|"warning"|"success"|null,
25         helpMsg?:string
26     }
27 }
28
29 /*
30 React component state interface, holds all data handled by the Form
31 */
32 export interface myState {
33     stage:number,
34     traveler?:Traveler,
35     travelType?:string,
36     travelInfo?:Array<string>,
37     validation?:FieldValidation
38 }

```

This App has a single form to collect and process user info, we are basing technology on the React-Bootstrap implementation

```

TS Main.ts TS CreatePNR.tsx X
src > code > TS CreatePNR.tsx > ...
38 }
39
40 /*
41 CreatePNR Component, multi-stage data entry form based on react-bootstrap component library
42 */
43 export class CreatePNR extends React.Component<{},myState>{
44
45     constructor(e){
46         super(e);
47
48         //bind event handlers to the component instance
49         this.handleChange = this.handleChange.bind(this);
50         this.executeService = this.executeService.bind(this);
51         this.closeAndRefresh = this.closeAndRefresh.bind(this);
52         this.goBack = this.goBack.bind(this);
53         this.goNext = this.goNext.bind(this);
54
55         //fill default state values during component initialization
56         this.state = {
57             stage:1,
58             traveler:{
59                 name:"",
60                 surname:"",
61                 typeCode:"ADT"
62             },
63             validation:{
64                 txtName:{isValid:false,status:null,helpMsg:null},
65                 txtSurname:{isValid:false,status:null,helpMsg:null},
66                 txtEmail:{isValid:false,status:null,helpMsg:null}
67             }
68         }
69     }
70 }
71
72 /*
73 Method to handle field change, perform validation and update state
74 */
75 handleChange(e){
76

```

React state holds data model which will used to collect data from the user and perform field validation

```

TS Main.ts TS CreatePNR.tsx X
src > code > TS CreatePNR.tsx > ...
72  /*
73  Method to handle field changes, perform validation and update state
74  */
75  handleChange(e){
76
77      const ctlId = e.target.id;
78      const fldValue = e.target.value;
79      const validationState = this.state.validation;
80
81      const tmpTraveler = this.state.traveler;
82      let tmpTravelType = this.state.travelType;
83
84      console.log("handleChange",ctlId,fldValue);
85
86      if(ctlId=== "txtName" || ctlId=== "txtSurname"){
87          const tmpValidation = validationState[ctlId]
88
89          const length = fldValue.length;
90          if(ctlId=== "txtName")
91              tmpTraveler.name = fldValue;
92          if(ctlId=== "txtSurname")
93              tmpTraveler.surname = fldValue;
94
95          if(length<=0){
96              tmpValidation.isValid = false;
97              tmpValidation.status = 'error';
98              tmpValidation.helpMsg = "required field";
99          }else if(length<=1){
100              tmpValidation.isValid = false;
101              tmpValidation.status = 'warning';
102              tmpValidation.helpMsg = "must be more than one character long";
103          }else if(length>1){
104              tmpValidation.isValid = true;
105              tmpValidation.status = 'success';
106              tmpValidation.helpMsg = null;
107
108          }
109      }
110  }

```

Field changes are controlled by handleChange method, here fields are validated and the React component state updated

1

TS Main.ts

TS CreatePNR.tsx

src > code > TS CreatePNR.tsx > CreatePNR > executeService

```

138
139  /*
140  Creates a UpdateReservationRQ request payload merging state data, then utilizes
141  SOAP API service handler to send the request and parse results
142  */
143  executeService(){
144      const soapApiService = getService(ISoapApiService);
145      const pl1 = `
146      <UpdateReservationRQ Version="1.19.8" xmlns="http://webservices.sabre.com/pnrbuilder/v1_19">
147      <RequestType commitTransaction="false" initialIgnore="true">Stateful</RequestType>
148      <ReturnOptions IncludeUpdateDetails="true" RetrievePNR="false"/>
149          <ReservationUpdateList>
150              <ReservationUpdateItem>
151                  <PassengerNameUpdate op="C">
152                      <TravelerName type="${this.state.traveler.typeCode}">
153                          <Given>${this.state.traveler.name}</Given>
154                          <Surname>${this.state.traveler.surname}</Surname>
155                      </TravelerName>
156                  </PassengerNameUpdate>
157              </ReservationUpdateItem>
158              <ReservationUpdateItem>
159                  <RemarkUpdate op="C">
160                      <RemarkText>THIS IS ${this.state.travelType} TRAVEL TYPE REMARK</RemarkText>
161                  </RemarkUpdate>
162              </ReservationUpdateItem>
163          </ReservationUpdateList>
164      </UpdateReservationRQ>
165      `;
166
167      soapApiService.callSws({action:"UpdateReservationRQ",payload:pl1,authTokenType:"SESSION"})
168      .then(
169          (res)=>{
170              //validate API response
171              console.log("Soap API call result",res);
172              if(res.errorCode || (res.value && res.value.indexOf("<st19:Error")>=0) ){
173                  this.setState({stage:4})
174              }
175              this.setState({stage:3})
176          }

```

Ln 180, Col 1 Spaces: 4 UTF-8 CRLF {} TypeScript React

Upon form submission Sabre SOAP API is called to write data to the current PNR on the user session.


```
File Edit Selection View Go Run Terminal Help CreatePNR.tsx - com-sabre-redapp-fundamentals-web-module - Visual Studio Code [Administrator]

TS Main.ts TS CreatePNR.tsx X

src > code > TS CreatePNR.tsx > CreatePNR > executeService
201 /*
202 Render parts of multi-stage form using react-bootstrap components
203 The data entry form is wrapped by a Modal Dialog component
204 */
205 render(): JSX.Element {
206
207     switch(this.state.stage){
208     case 1:
209         const validateName = this.state.validation["txtName"];
210         const validateSurname = this.state.validation["txtSurname"];
211         return (
212             <Modal.Dialog className="react-modal">
213                 <Modal.Header closeButton onHide={()={this.handleModalClose();}}>
214                     <Modal.Title>Data Entry Form (1 of 2)</Modal.Title>
215                 </Modal.Header>
216                 <Modal.Body>
217                     <Form autoComplete="off">
218                         <Panel>
219                             <Panel.Heading>
220                                 <Panel.Title>About Traveler</Panel.Title>
221                             </Panel.Heading>
222                             <Panel.Body>
223                                 <FormGroup controlId="txtName" validationState={validateName.status}>
224                                     <ControllLabel>Name</ControllLabel>
225                                     <FormControl
226                                         type="text"
227                                         placeholder="Enter traveler Name"
228                                         value={this.state.traveler.name}
229                                         onChange={this.handleChange} />
230                                     {validateName.helpMsg && <FormControl.Feedback />}
231                                     {(validateName.helpMsg) && <HelpBlock>{validateName.helpMsg}</HelpBlock>}
232                                 </FormGroup>
233
234                                 <FormGroup controlId="txtSurname" validationState={validateSurname.status}>
235                                     <ControllLabel>Surname</ControllLabel>
236                                     <FormControl
237                                         type="text"
238                                         placeholder="Enter traveler Surname"
239                                         value={this.state.traveler.surname}
240                                         onChange={this.handleChange} />
241                                     {validateSurname.helpMsg && <FormControl.Feedback />}
242                                     {(validateSurname.helpMsg) && <HelpBlock>{validateSurname.helpMsg}</HelpBlock>}
243                                 </FormGroup>
244                             </Panel.Body>
245                         </Panel>
246                     </Form>
247                 </Modal.Body>
248             </Modal.Dialog>
249         );
250     }
251 }
```

This react component is based entirely on the React-bootstrap (v3) library available on the web SDK, more info could be found @ <https://react-bootstrap-v3-netlify.app/getting-started/introduction>

```

TS Main.ts TS CreatePNR.tsx X
src > code > TS CreatePNR.tsx > CreatePNR > executeService
264
265     case 2:
266         return (
267             <Modal.Dialog className="react-modal">
268                 <Modal.Header closeButton onHide={()=>{this.handleModalClose();}}>
269                     <Modal.Title>Data Entry Form (2 of 2)</Modal.Title>
270                 </Modal.Header>
271                 <Modal.Body>
272                     <Form>
273                         <Panel>
274                             <Panel.Heading><Panel.Title>About Travel</Panel.Title></Panel.Heading>
275                             <Panel.Body>
276                                 <FormGroup controlId="selTravelType">
277                                     <ControllLabel>Travel Type</ControllLabel>
278                                     <FormControl className="select" placeholder="select" onChange={this.handleChange} value={this.state.travelType}>
279                                         <option value="select">select</option>
280                                         <option value="business">business</option>
281                                         <option value="leisure">leisure</option>
282                                     </FormControl>
283                                 </FormGroup>
284                                 { this.state.travelType==="business" &&
285                                     <FormGroup>
286                                         <ControllLabel>Add Corporate ID ?</ControllLabel>
287                                         <InputGroup>
288                                             <InputGroup.Addon>
289                                                 <input type="checkbox" aria-label="..." />
290                                             </InputGroup.Addon>
291                                             <FormControl type="text" />
292                                         </InputGroup>
293                                     </FormGroup>
294                                 }
295                                 { this.state.travelType==="leisure" &&
296                                     <FormGroup>
297                                         <ControllLabel>Add Special Service Request ?</ControllLabel>
298                                         <InputGroup>
299                                             <InputGroup.Addon>
300                                                 <input type="checkbox" aria-label="..." />
301                                             </InputGroup.Addon>
302                                         <FormControl type="text" />

```

Multi-stage form with validation, implementation details

```

TS Main.ts TS CreatePNR.tsx X
src > code > TS CreatePNR.tsx > CreatePNR > executeService
309
310     </Form>
311   </Modal.Body>
312   <Modal.Footer>
313     <Button onClick={this.handleModalClose} className="btn btn-secondary">Cancel</Button>
314     <Button className="btn btn-primary" onClick={this.goBack}>Back</Button>
315     <Button className="btn btn-primary btn-success" onClick={this.executeService}>Create PNR</Button>
316
317   </Modal.Footer>
318 </Modal.Dialog>
319 );
320 case 3:
321   return(
322     <Alert bsStyle="success" onDismiss={this.closeAndRefresh}>
323       <h4>Success</h4>
324       <hr/>
325       <p>Operation completed sucessfully, data was written to the PNR, session status will be refreshed...</p>
326       <hr/>
327       <p>
328         <Button bsStyle="success" onClick={this.closeAndRefresh}>Close</Button>
329       </p>
330     </Alert>
331   );
332 case 4:
333   return(
334     <Alert bsStyle="danger" onDismiss={this.handleModalClose}>
335       <h4>Error</h4>
336       <hr/>
337       <p>
338         The operation could not be completed, validate records and try again...
339       </p>
340       <hr/>
341       <p>
342         <Button bsStyle="danger" onClick={this.goBack}>Retry</Button>
343         <Button onClick={this.handleModalClose}>Cancel</Button>
344       </p>
345     </Alert>
346   );
347

```

Alerts are used to inform success or failure of the operation

```

src > code > TS Main.ts > Main > showForm
1  import {Module} from 'sabre-ngv-core/modules/Module';
2  import { getService, registerService } from './Context';
3  import { ExtensionPointService } from "sabre-ngv-xp/services/ExtensionPointService";
4  import { RedAppSidePanelConfig } from "sabre-ngv-xp/configs/RedAppSidePanelConfig";
5  import { RedAppSidePanelButton } from "sabre-ngv-redAppSidePanel/models/RedAppSidePanelButton";
6  import { LayerService } from "sabre-ngv-core/services/LayerService";
7  import { CreatePNR } from './CreatePNR';
8
9  export class Main extends Module {
10
11      /**
12       * Web module initializaiton, configure menu contributions and register helper service
13       */
14      init(): void {
15          super.init();
16          // get reference to ExtensionPointService, which is used to add a new actionable item to the Custom Workflow panel (sidebar)
17          const xp = getService(ExtensionPointService);
18          //create a new sidebar contribution configuration
19          const sidepanelMenu = new RedAppSidePanelConfig(
20              [new RedAppSidePanelButton(
21                  "Create PNR",

```

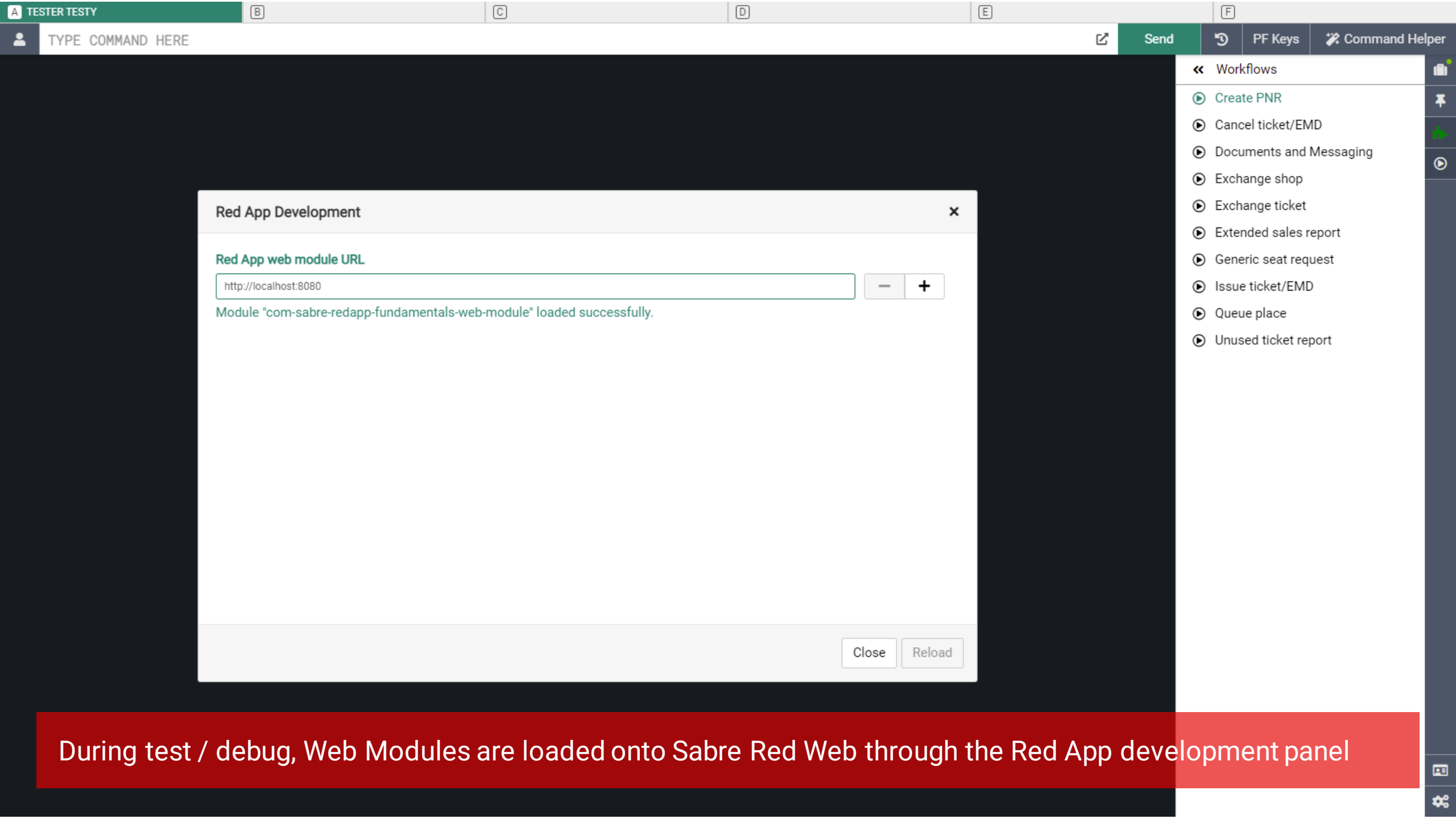
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```

[C860] 10:52:24 .. -> build:dist -> ..
[C860] 10:52:24 .. -> build:dist:source-maps -> ..
[C860] 10:52:24 .. -> build:dist:source-maps::atom -> ..
[C860] 10:52:25 .. -> build:dist::atom -> ..
[C860] 10:52:25 > cp C:\...\OneDrive - Sabre\Dev\Developer Workbench\Fundamentals\code\web-src\com-sabre-redapp-fundamentals-web-module\build\prod C:\...\OneDrive - Sabre\Dev\Developer Workbench\Fundamentals\code\web-src\com-sabre-redapp-fundamentals-web-module\build\dist
[C860] 10:52:25 Delivering prod to C:\...\OneDrive - Sabre\Dev\Developer Workbench\Fundamentals\code\web\com-sabre-redapp-fundamentals-web-module
[C860] 10:52:25 > cp C:\...\OneDrive - Sabre\Dev\Developer Workbench\Fundamentals\code\web-src\com-sabre-redapp-fundamentals-web-module\build\prod C:\...\OneDrive - Sabre\Dev\Developer Workbench\Fundamentals\code\web\com-sabre-redapp-fundamentals-web-module
[C860] 10:52:26 .. -> run:watch -> ..
[C860] 10:52:26 .. -> run:watch::atom -> ..
[C860] 10:52:26 [WATCH] ..... READY! .....
[C860] 10:52:26 .. -> run:server -> ..
[C860] 10:52:26 .. -> run:server::atom -> ..
[C860] 10:52:26 http://localhost:8080

```

During implementation and test, NGV run command is used to host web-module to be loaded on Sabre Red Web instance.



Red App Development

Red App web module URL

http://localhost:8080

Module "com-sabre-redapp-fundamentals-web-module" loaded successfully.

Close

Reload

- Workflows
- Create PNR

Cancel ticket/EMD

Documents and Messaging

Exchange shop

Exchange ticket

Extended sales report

Generic seat request

Issue ticket/EMD

Queue place

Unused ticket report

During test / debug, Web Modules are loaded onto Sabre Red Web through the Red App development panel

Sabre®