Field Work Maintenance App

Author: Juan Alejandro Knight

1. Story

The goal of this application is to allow field workers to view pending orders and make much easier the programming of maintenance in different geographical locations. The worker will be able to take decisions according to the order's type, priority and proximity to current location. The application is designed for and oil and gas company that has pipelines across different cities. The application will be accessible through mobile devices to keep track of the open orders.

The segmentation is focused to any industry that has needs a specialized work taskforce to maintain facilities or company's assets on different locations, optimizing business needs and traveling efficiency.

The targeting of the application is designed for the taskforce leader or programmer, making decisions easier and with the appropriate information. The specific goal of the app is to program when will each order be executed and the estimated time of work and travel.

To be able to position the app, the programming must be accessible through mobile devices on remote locations, to speed up processes and avoid bottlenecks. The application is integrated with Google Maps to facilitate geospatial analysis and location's traveling distance calculus. The end-user does not have a global business picture or has skills to work with complex systems, so the app must be simple and to the point.

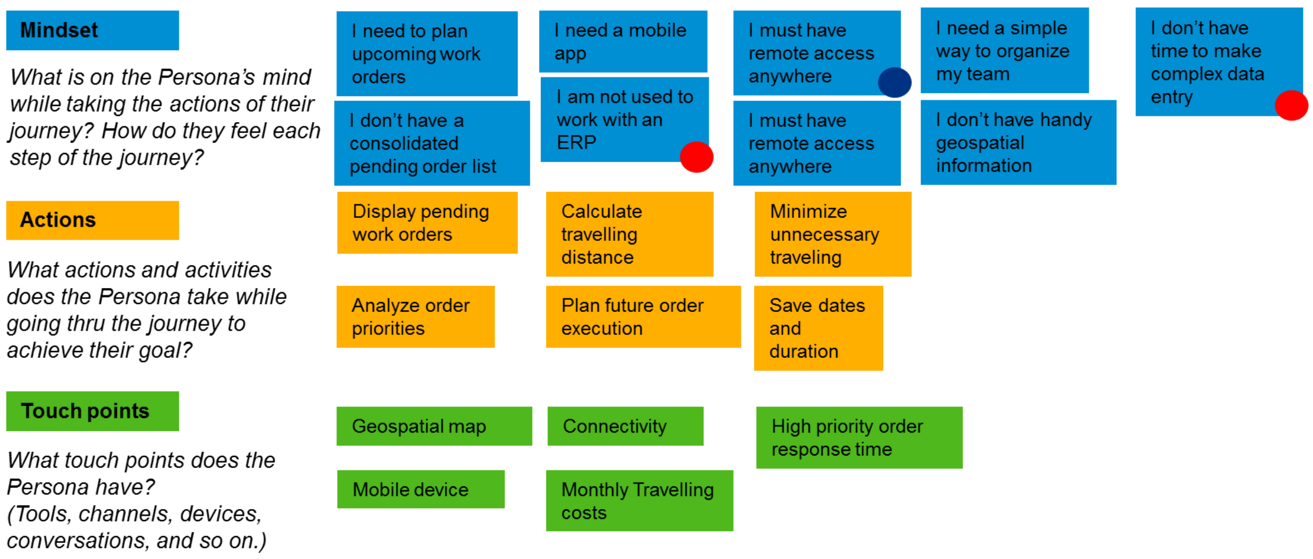
2. Persona

- Name: John
- Job title: Field maintenance team leader
- Background: 35 years old, has worked out in the field all his life. He is not comfortable using computers or complex software. He leads a team of 3 workers and repairs pipelines across different locations. He is accustomed to travel and work on different locations. He does not have a business overview, and is very focused on day-to-day tasks.
- Job responsibilities:
  - Coordinate a team of field workers to maintain oil and gas pipelines
  - Track pending work orders
  - Program the team's work on different locations
  - Inform the work orders remediation status
  - Prioritize order execution
- Objective: I would like to make my team work effectively and efficiently to minimize production downtime.
- Main goals:
  - Reduce production downtime
  - Keep the team occupied effectively
  - Minimize travelling effort, time and expenses
  - Minimize risks with proactive maintenance
- Needs:

    - I need to coordinate easily my travelling and order execution in different locations
    - I need relevant information to prioritize which orders are solved first.
    - I need to minimize the time of unsolved high priority orders.
    - I need to program dates and task durations.
- Pain points:
    - Difficult to understand complex systems
    - Not enough information to take decisions
    - No inbox to keep track of pending orders
    - No mobile access to order programming
- Stakeholders:
    - Field taskforce members
    - Maintenance officer
    - Plant manager
- Competences
    - Casual vs Power user: Extremely casual user, has little or no IT skills.
    - Proactive vs Reactive: The urgent tasks are purely reactive since high priority orders need reaction. Preventive maintenance requires proactiveness.
    - Group or Individual work: Definitely works in team.
    - Focus: Local focus to toy-to-day tasks. No global picture
    - Innovative vs Conservative: Very conservative. Has been working in the same way for a very long time. Is not prone to disruptive innovation.

3. User Experience Journey

The UX Journey depends on the end-user's mindset, actions and touch points:



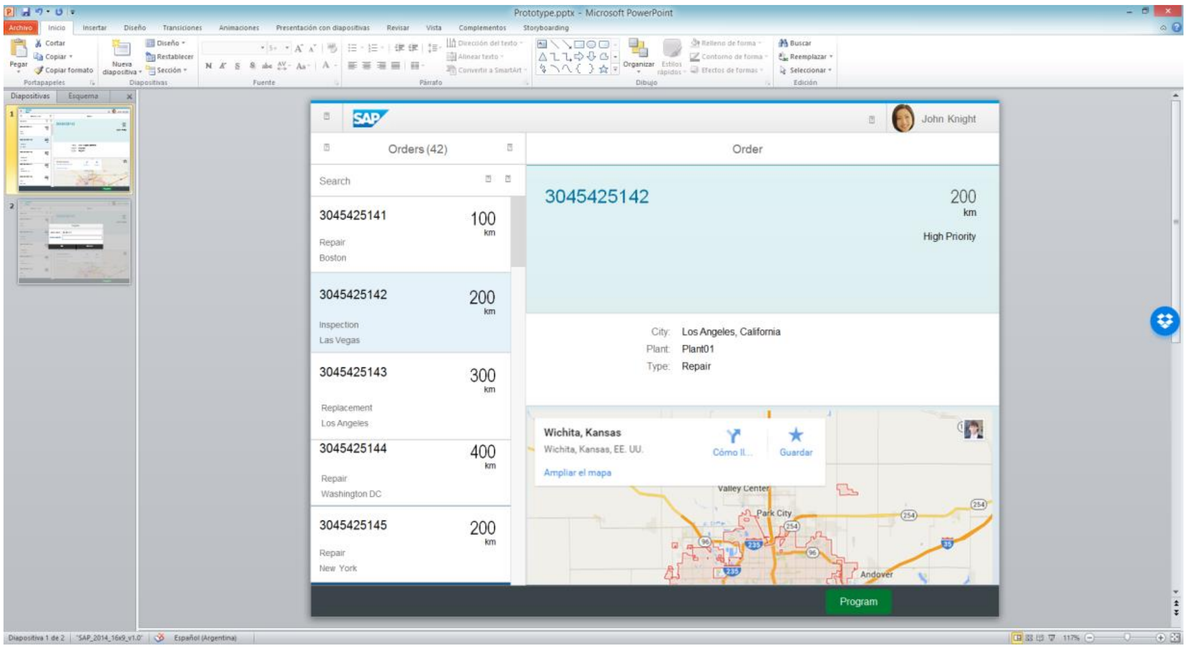
Mindset *What is on the Persona's mind while taking the actions of their journey? How do they feel each step of the journey?*	I need to plan upcoming work orders; I don't have a consolidated pending order list	I need a mobile app; I am not used to work with an ERP	I must have remote access anywhere; I must have remote access anywhere	I need a simple way to organize my team; I don't have handy geospatial information	I don't have time to make complex data entry
Actions *What actions and activities does the Persona take while going thru the journey to achieve their goal?*	Display pending work orders; Analyze order priorities	Calculate travelling distance; Plan future order execution	Minimize unnecessary traveling; Save dates and duration		
Touch points *What touch points does the Persona have? (Tools, channels, devices, conversations, and so on.)*	Geospatial map; Mobile device	Connectivity; Monthly Travelling costs	High priority order response time		

4. Mock up

I used the SAP Fiori Prototyping Kit in Microsoft PowerPoint to design the screen prototypes according to SAP Fiori Guidelines.

The application responds to a master/detail design. In the first screen, the user can track pending work orders. When clicking an order in the master view, the details will be pulled from SAP, showing additional information.

A Google Map is rendered to pin point the work order’s location and calculate travelling distance.

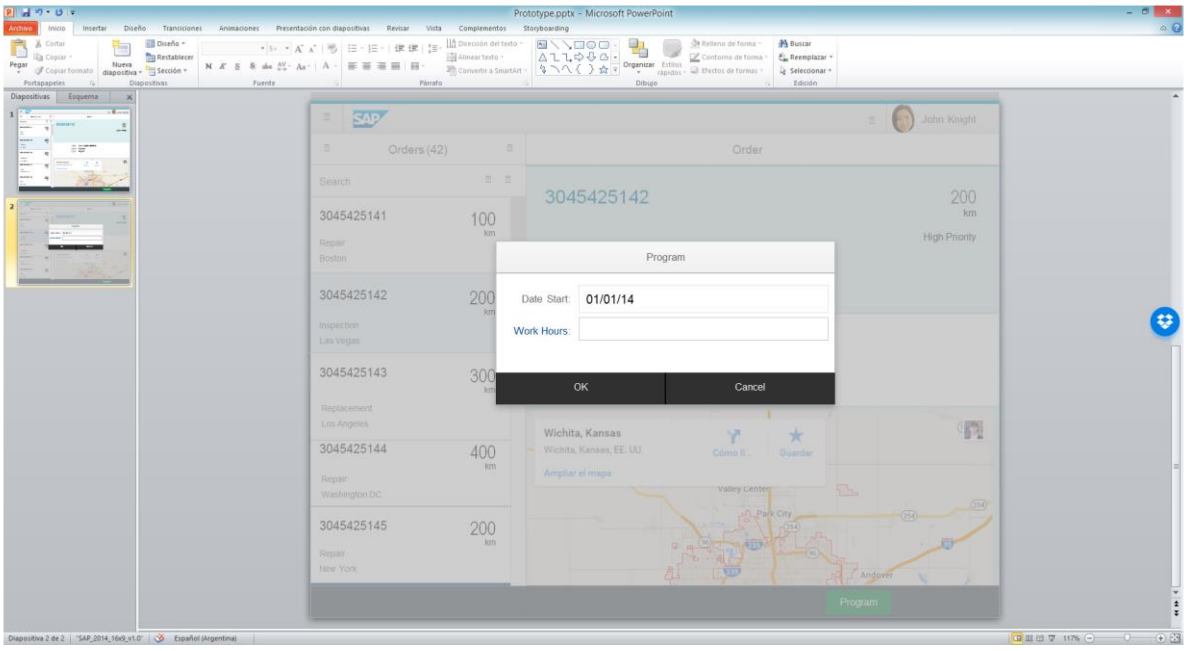


3045425141	100 km	Repair	Boston
3045425142	200 km	Inspection	Las Vegas
3045425143	300 km	Replacement	Los Angeles
3045425144	400 km	Repair	Washington DC
3045425145	200 km	Repair	New York

3045425142 200 km
High Priority

City: Los Angeles, California
Plant: Plant01
Type: Repair

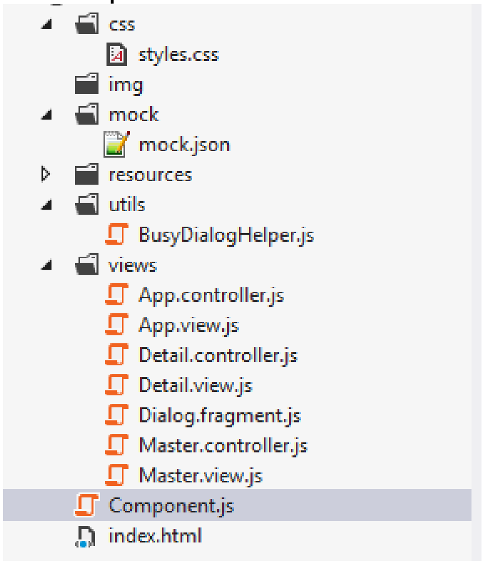
The user can select an order and hit the “Program” button, allowing him to select on which day will the team arrive and how many hours will he spend on the order. The program will then be saved on SAP ERP.



Date Start: 01/01/14
Work Hours:

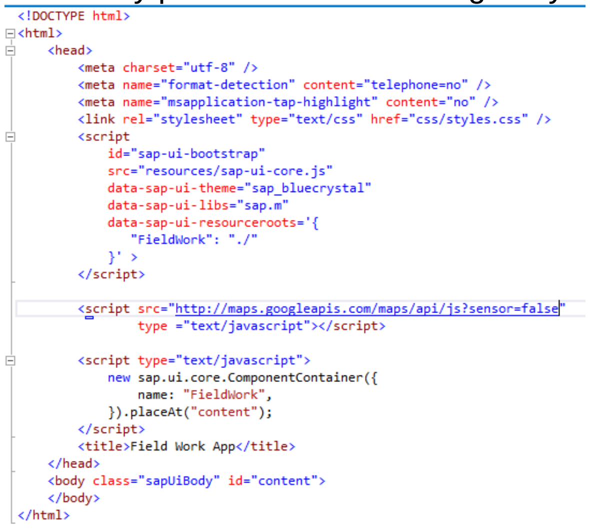
5. WebIDE Screenshots + Video

The structure of the app is quite simple:



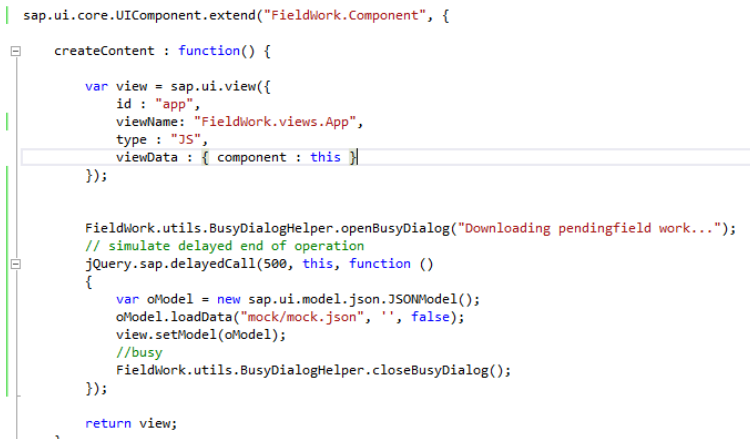
```
◢ css
    styles.css
  img
◢ mock
    mock.json
▷ resources
◢ utils
    BusyDialogHelper.js
◢ views
    App.controller.js
    App.view.js
    Detail.controller.js
    Detail.view.js
    Dialog.fragment.js
    Master.controller.js
    Master.view.js
  Component.js
  index.html
```

- Index.html: Application's entry point accessible through any HTML5 browser.



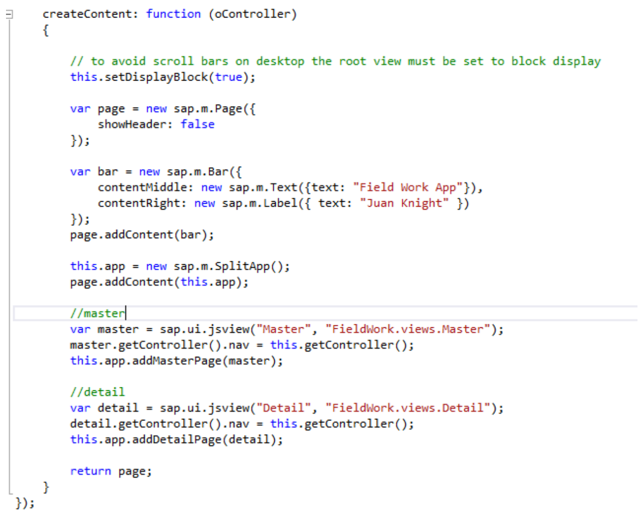
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <meta name="format-detection" content="telephone=no" />
        <meta name="msapplication-tap-highlight" content="no" />
        <link rel="stylesheet" type="text/css" href="css/styles.css" />
        <script
            id="sap-ui-bootstrap"
            src="resources/sap-ui-core.js"
            data-sap-ui-theme="sap_bluecrystal"
            data-sap-ui-libs="sap.m"
            data-sap-ui-resourceroots='{
               "FieldWork": "./"
            }' >
        </script>

        <script src="http://maps.googleapis.com/maps/api/js?sensor=false"
                type ="text/javascript"></script>

        <script type="text/javascript">
            new sap.ui.core.ComponentContainer({
                name: "FieldWork",
            }).placeAt("content");
        </script>
        <title>Field Work App</title>
    </head>
    <body class="sapUiBody" id="content">
    </body>
</html>
```

- Component: Basic application startup and settings.



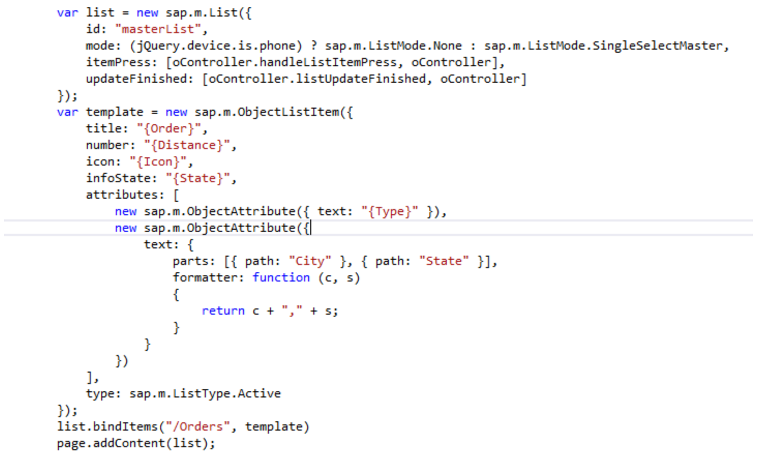
```
sap.ui.core.UIComponent.extend("FieldWork.Component", {

    createContent : function() {

        var view = sap.ui.view({
            id : "app",
            viewName: "FieldWork.views.App",
            type : "JS",
            viewData : { component : this }
        });


        FieldWork.utils.BusyDialogHelper.openBusyDialog("Downloading pendingfield work...");
        // simulate delayed end of operation
        jQuery.sap.delayedCall(500, this, function ()
        {
            var oModel = new sap.ui.model.json.JSONModel();
            oModel.loadData("mock/mock.json", '', false);
            view.setModel(oModel);
            //busy
            FieldWork.utils.BusyDialogHelper.closeBusyDialog();
        });

        return view;
```

- App view: Encapsulates de sap.m.SplitApp control to navigate between master and detail pages.



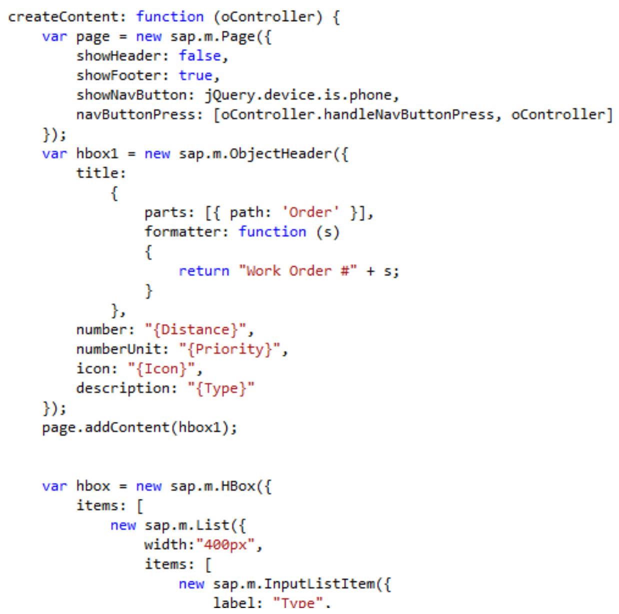
```
createContent: function (oController)
{

    // to avoid scroll bars on desktop the root view must be set to block display
    this.setDisplayBlock(true);

    var page = new sap.m.Page({
        showHeader: false
    });

    var bar = new sap.m.Bar({
        contentMiddle: new sap.m.Text({text: "Field Work App"}),
        contentRight: new sap.m.Label({ text: "Juan Knight" })
    });
    page.addContent(bar);

    this.app = new sap.m.SplitApp();
    page.addContent(this.app);

    //master
    var master = sap.ui.jsview("Master", "FieldWork.views.Master");
    master.getController().nav = this.getController();
    this.app.addMasterPage(master);

    //detail
    var detail = sap.ui.jsview("Detail", "FieldWork.views.Detail");
    detail.getController().nav = this.getController();
    this.app.addDetailPage(detail);

    return page;
}
});
```

- Master view: Shows the list of pending orders and the key information to select the order. The items are styled according to the orders' priorities



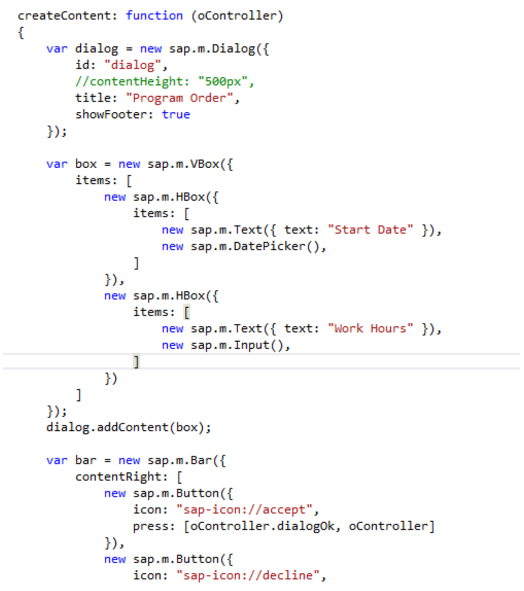
```
var list = new sap.m.List({
    id: "masterList",
    mode: (jQuery.device.is.phone) ? sap.m.ListMode.None : sap.m.ListMode.SingleSelectMaster,
    itemPress: [oController.handleListItemPress, oController],
    updateFinished: [oController.listUpdateFinished, oController]
});
var template = new sap.m.ObjectListItem({
    title: "{Order}",
    number: "{Distance}",
    icon: "{Icon}",
    infoState: "{State}",
    attributes: [
        new sap.m.ObjectAttribute({ text: "{Type}" }),
        new sap.m.ObjectAttribute({
            text: {
                parts: [{ path: "City" }, { path: "State" }],
                formatter: function (c, s)
                {
                    return c + "," + s;
                }
            }
        })
    ],
    type: sap.m.ListType.Active
});
list.bindItems("/Orders", template)
page.addContent(list);
```

- Detail view: shows extended order information and a Google Map with the remote location.



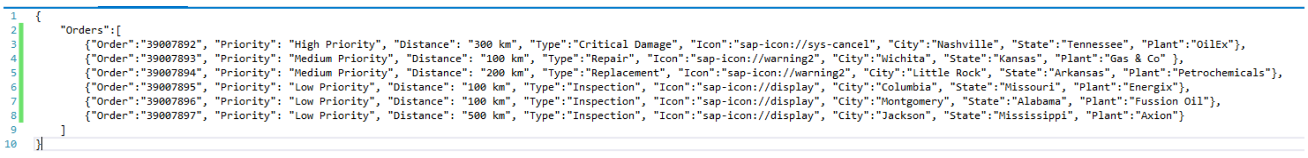
```
createContent: function (oController) {
    var page = new sap.m.Page({
        showHeader: false,
        showFooter: true,
        showNavButton: jQuery.device.is.phone,
        navButtonPress: [oController.handleNavButtonPress, oController]
    });
    var hbox1 = new sap.m.ObjectHeader({
        title:
            {
                parts: [{ path: 'Order' }],
                formatter: function (s)
                {
                    return "Work Order #" + s;
                }
            },
        number: "{Distance}",
        numberUnit: "{Priority}",
        icon: "{Icon}",
        description: "{Type}"
    });
    page.addContent(hbox1);


    var hbox = new sap.m.HBox({
        items: [
            new sap.m.List({
                width:"400px",
                items: [
                    new sap.m.InputListItem({
                        label: "Type",
```

- Dialog fragment: enable the "Program" dialog:



```
createContent: function (oController)
{
    var dialog = new sap.m.Dialog({
        id: "dialog",
        //contentHeight: "500px",
        title: "Program Order",
        showFooter: true
    });

    var box = new sap.m.VBox({
        items: [
            new sap.m.HBox({
                items: [
                    new sap.m.Text({ text: "Start Date" }),
                    new sap.m.DatePicker(),
                ]
            }),
            new sap.m.HBox({
                items: [
                    new sap.m.Text({ text: "Work Hours" }),
                    new sap.m.Input(),
                ]
            })
        ]
    });
    dialog.addContent(box);

    var bar = new sap.m.Bar({
        contentRight: [
            new sap.m.Button({
                icon: "sap-icon://accept",
                press: [oController.dialogOk, oController]
            }),
            new sap.m.Button({
                icon: "sap-icon://decline",
```

The app was built using a SplitApp for a Master/Detail template. The mock data generated to emulate a live backend system.

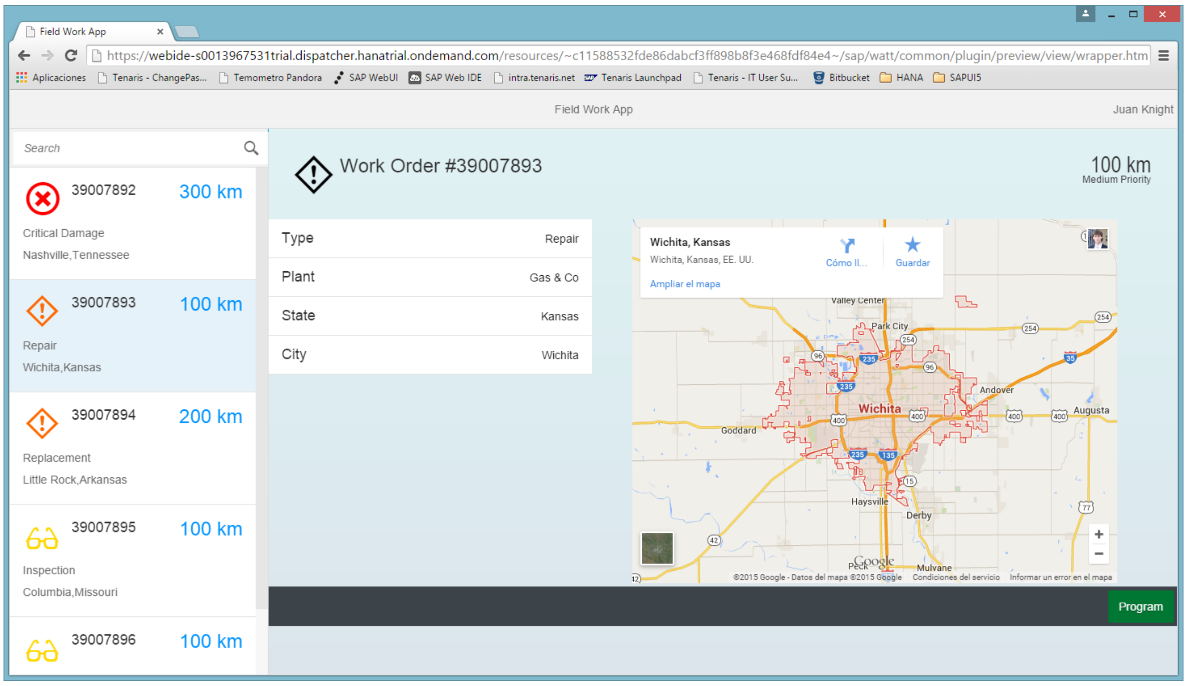


```
{
    "Orders":[
        {"Order":"39007892", "Priority": "High Priority", "Distance": "300 km", "Type":"Critical Damage", "Icon":"sap-icon://sys-cancel", "City":"Nashville", "State":"Tennessee", "Plant":"OilEx"},
        {"Order":"39007893", "Priority": "Medium Priority", "Distance": "100 km", "Type":"Repair", "Icon":"sap-icon://warning2", "City":"Wichita", "State":"Kansas", "Plant":"Gas & Co" },
        {"Order":"39007894", "Priority": "Medium Priority", "Distance": "200 km", "Type":"Replacement", "Icon":"sap-icon://warning2", "City":"Little Rock", "State":"Arkansas", "Plant":"Petrochemicals"},
        {"Order":"39007895", "Priority": "Low Priority", "Distance": "100 km", "Type":"Inspection", "Icon":"sap-icon://display", "City":"Columbia", "State":"Missouri", "Plant":"Energix"},
        {"Order":"39007896", "Priority": "Low Priority", "Distance": "100 km", "Type":"Inspection", "Icon":"sap-icon://display", "City":"Montgomery", "State":"Alabama", "Plant":"Fussion Oil"},
        {"Order":"39007897", "Priority": "Low Priority", "Distance": "500 km", "Type":"Inspection", "Icon":"sap-icon://display", "City":"Jackson", "State":"Mississippi", "Plant":"Axion"}
    ]
}
```

The orders on the master view have been styled to make high-priority orders easily detectable using icons and font colors to distinguish the urgency of different orders. This empowers the user to focus on the orders

When clicking on a list item, the detail page pulls the data for the selected order and renders a Google Map pinpointing the location.

A footer was added to enable the user to hit the Program Button using conventional button colors for an “approve/submit” button.



Field Work App

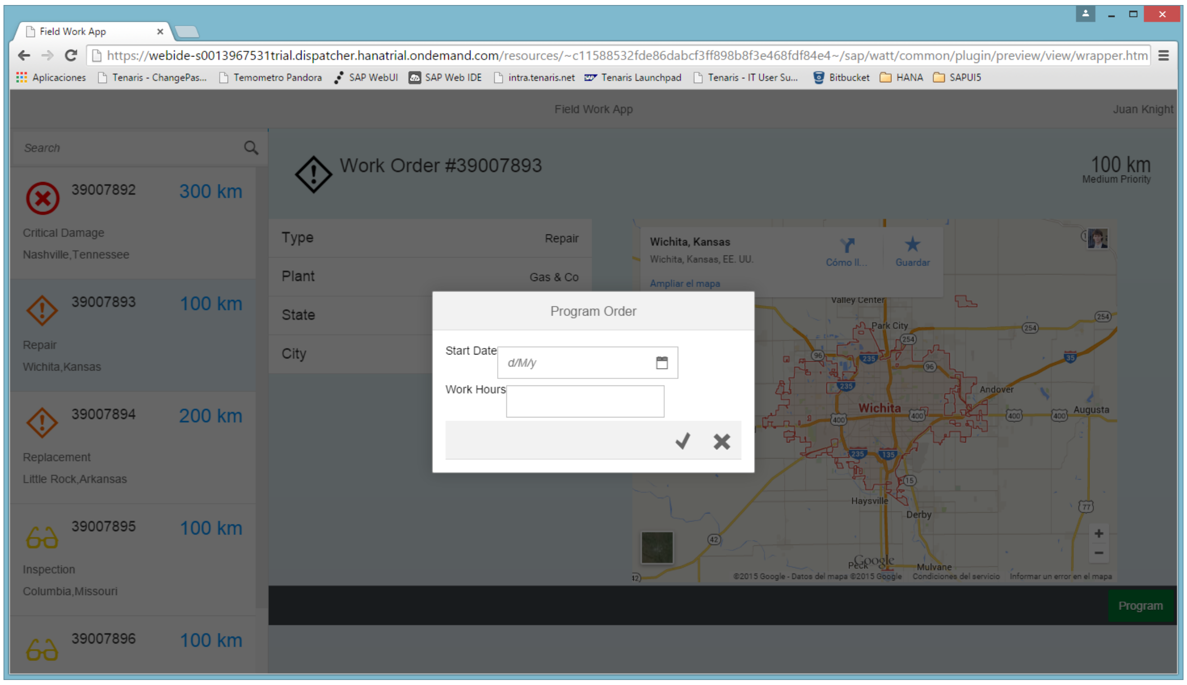
39007892	300 km	Critical Damage, Nashville,Tennessee
39007893	100 km	Repair, Wichita,Kansas
39007894	200 km	Replacement, Little Rock,Arkansas
39007895	100 km	Inspection, Columbia,Missouri
39007896	100 km	

Work Order #39007893

100 km
Medium Priority

Type	Repair
Plant	Gas & Co
State	Kansas
City	Wichita

When clicking the Program Button, the user must type in the date and working hours in his plan.



Program Order

Start Date	d/M/y
Work Hours	