How to create PDF forms in Xamarin.Forms applications

Written by Apitron Documentation Team

Introduction

Xamarin.Forms might be the best fit technology for cross-platform forms data processing applications as its name suggests. Nowadays we have lots of forms to fill, and many apps emerge to help us with that. In many cases, we also need these forms to be exported in some way or read afterwards and the PDF became the de-facto standard.

Luckily, the PDF standard offers such things like *fields* – document properties that can be added, saved, read and modified during the document's lifetime. Furthermore, they can be interactively edited using PDF viewers, being linked to corresponding widget annotation (see the section 12.7 *Interactive Forms* of the PDF specification for the details).

In this article, we'll show you how to create a simple PDF form using Xamarin.Forms and Apitron PDF Kit .NET component.

Form layout and code

We'll use very simple layout – a button to trigger the saving action and a few text boxes for entering the data of an imaginable employee. See the XAML and code behind below:

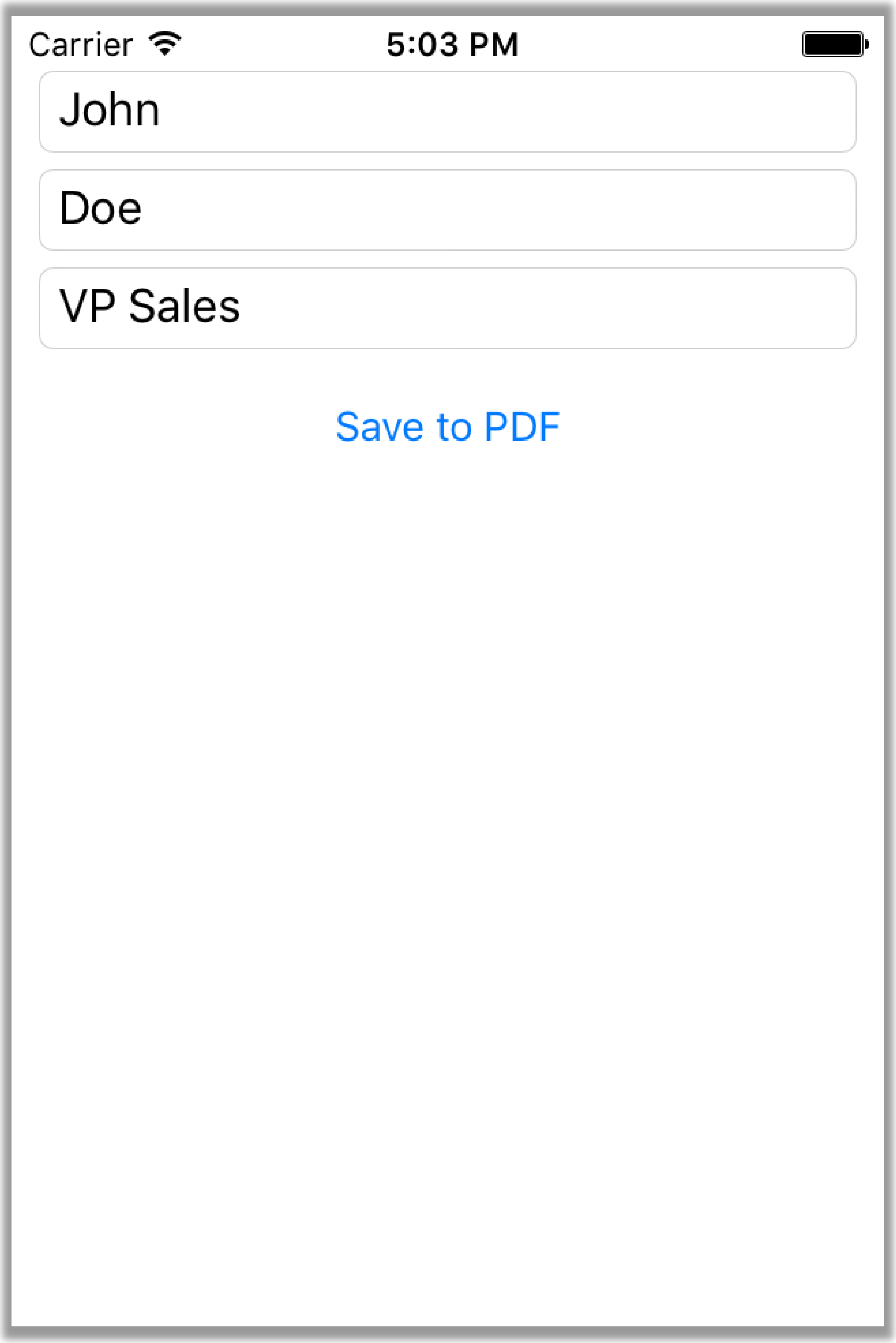
```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="PDFFormCreationSample.MyPage">
        <StackLayout Orientation="Vertical" Padding="10,20,10,10">
            <Entry x:Name="firstName" Text="{Binding FirstName, Mode=TwoWay}"
                Placeholder="Enter employee's first name here"/>
            <Entry x:Name="lastName" Text="{Binding LastName, Mode=TwoWay}"
                Placeholder="Enter employee's last name here"/>
            <Entry x:Name="position" Text="{Binding CurrentPosition, Mode=TwoWay}"
                Placeholder="Enter employee's position here"/>
            <Button Text="Save to PDF" Clicked="OnSaveClicked" />
        </StackLayout>
</ContentPage>
```

```
public partial class MyPage : ContentPage
{
    Employee currentEmployee;

    public MyPage ()
    {
        InitializeComponent ();
        BindingContext = currentEmployee = new Employee ();
    }
    public void OnSaveClicked(object sender, EventArgs args)
    {
        // create flow document and register necessary styles
        FlowDocument doc = new FlowDocument(){ Margin = new Thickness (10,10,10,10)};
        // the style for all document's textblocks and textboxes
        doc.StyleManager.RegisterStyle("TextBlock, TextBox", new Style() {
                Font = new Font("Helvetica",20),
                Color = RgbColors.Black, Display = Display.Block
            });

        // the style for the section that contains employee data
        doc.StyleManager.RegisterStyle ("#border", new Style (){
                Padding = new Thickness(10,10,10,10),
                BorderColor = RgbColors.DarkRed,Border = new Border(5), BorderRadius=5
            }
        );

        // add PDF form fields for later processing
        doc.Fields.Add (new TextField ("firstName", currentEmployee.FirstName));
        doc.Fields.Add (new TextField ("lastName", currentEmployee.LastName));
        doc.Fields.Add (new TextField ("position", currentEmployee.CurrentPosition));
        // create section and add text block inside
        Section section = new Section (){Id="border"};

    //  ios PDF preview doesn't display text fields correctly,
    //  uncomment this code to use simple text blocks instead of text boxes
    // section.Add(new TextBlock(string.Format("First name: {0}",currentEmployee.FirstName)));
    // section.Add(new TextBlock(string.Format("Last name: {0}", currentEmployee.LastName)));
    //section.Add(new TextBlock(string.Format("Position: {0}",currentEmployee.CurrentPosition )));

        section.Add(new TextBlock("First name: "));
        section.Add(new TextBox("firstName"));
        section.Add(new TextBlock("Last name: "));
        section.Add(new TextBox("lastName"));
        section.Add(new TextBlock("Position: "));
        section.Add(new TextBox("position"));
        doc.Add (section);

        // get io service and generate output file path
        var fileManager = DependencyService.Get<IFileIO>();
        string filePath = Path.Combine (fileManager.GetMyDocumentsPath (), "form.pdf");

        // generate document
        using(Stream outputStream = fileManager.CreateFile(filePath))
        {
            doc.Write (outputStream, new ResourceManager());
        }
        // request preview
        DependencyService.Get<IPDFPreviewProvider>().TriggerPreview (filePath);
    }
}
```

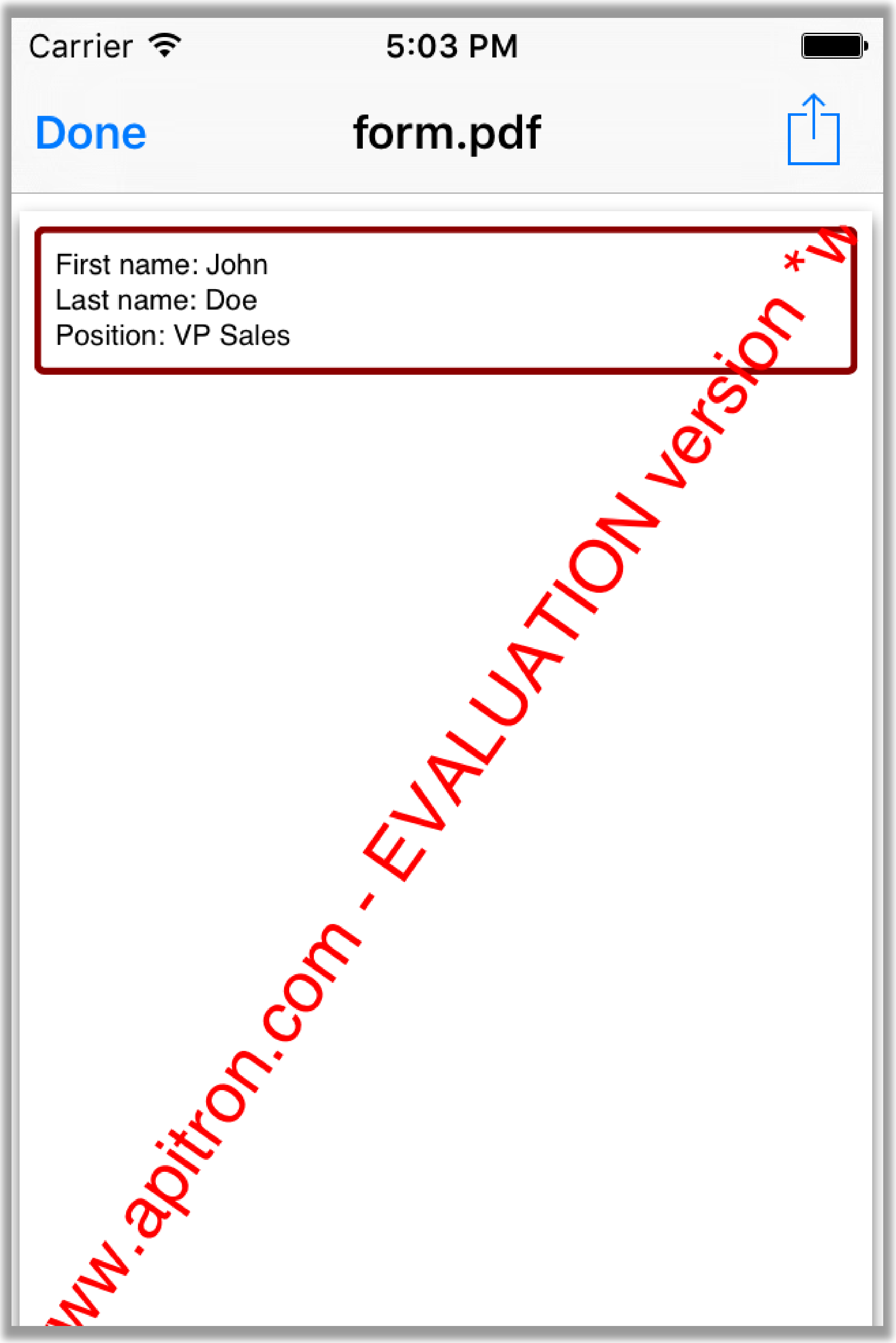
We used flow layout API to create the PDF form, and, unfortunately, as built-in iOS PDF preview doesn't display the PDF forms correctly, we've used simple text blocks instead of text fields (see the commented lines) for demoing on iOS. We also used *DependencyService* to request the platform specific IO and preview functionality.

Results

PDF forms generated by the iOS and Android versions are shown on the images below:

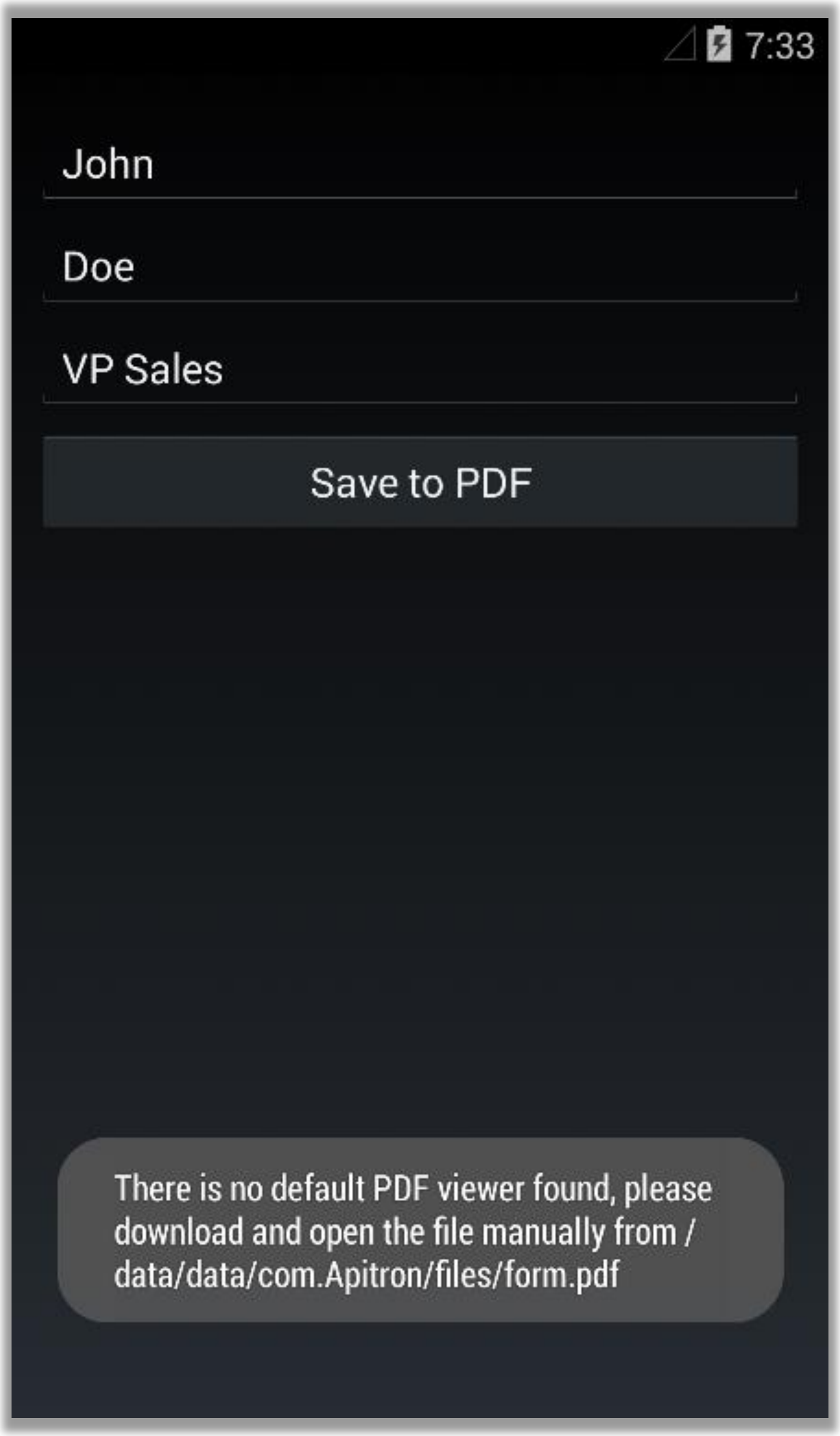


Pic. 1 Create PDF form using Xamarin.Forms (iOS UI)



First name: John
Last name: Doe
Position: VP Sales

Pic. 2 Created PDF form (iOS preview)


John

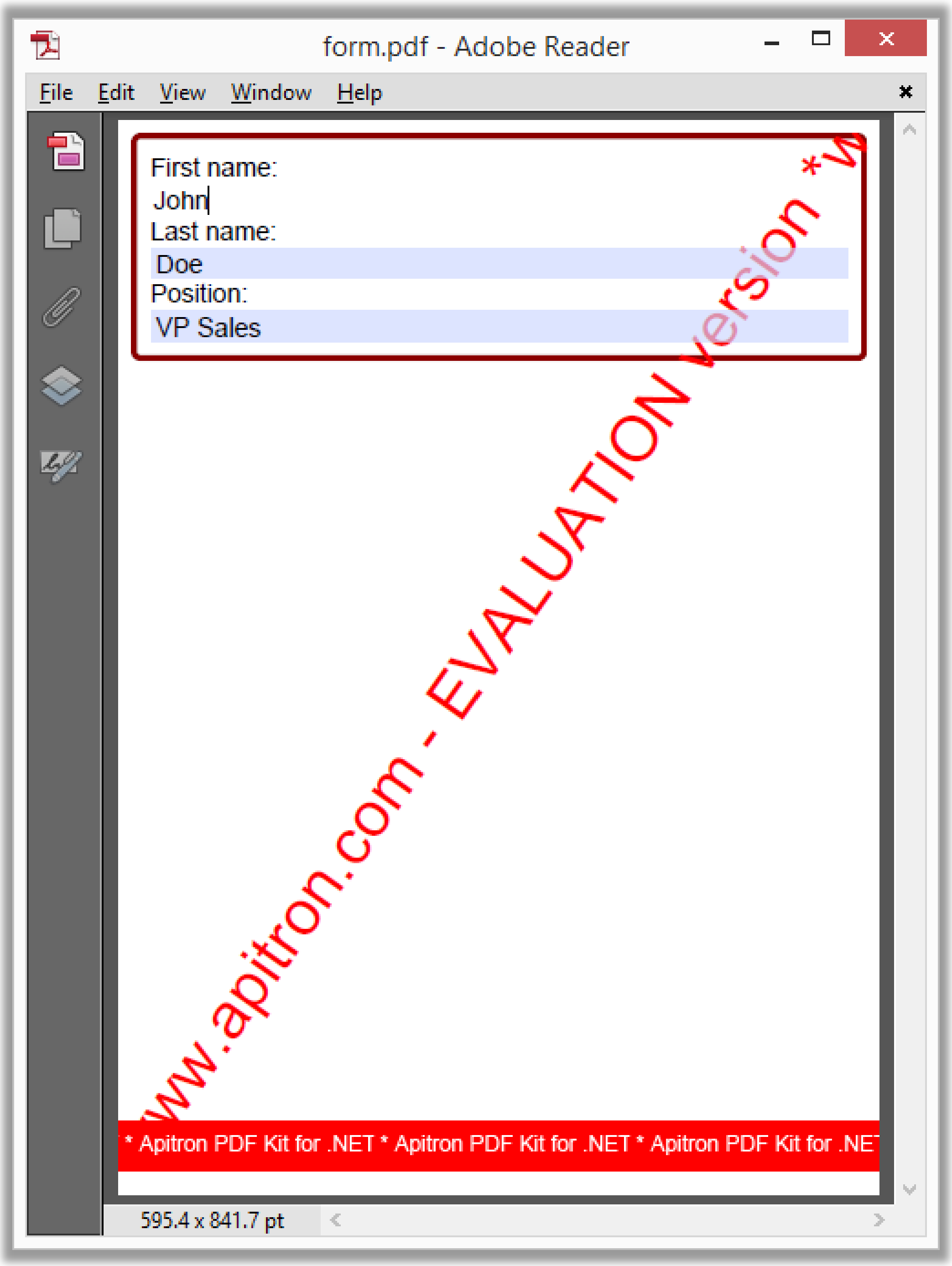
Doe

VP Sales

There is no default PDF viewer found, please download and open the file manually from /data/data/com.Apitron/files/form.pdf

Pic. 3 Create PDF form using Xamarin.Forms (Android UI)

If there is no default viewer found on your Android device, you may navigate to the indicated folder, copy the generated PDF form to the PC and view it using installed PDF viewer.



First name:
John
Last name:
Doe
Position:
VP Sales

Pic. 4 PDF form generated by the Xamarin.Forms app (Android)

On this image you can see that form fields are represented by editable text boxes and can be changed after the form creation. It's possible to change this behavior by making the fields read-only.

Conclusion

Apitron PDF Kit is fully compatible with Xamarin.Forms and can be used for creation of data-entry and data processing apps that require PDF export of import functionality. It’s also available for other platforms offering the same API for .NET applications running on desktops, windows phones, tablets and even web servers. You can download it by the following link. The complete example can be found in our GitHub repo.