Blazor | Electron | .NET | C# | TypeScript | Angular

windows .developer

windows .developer

6.2021

www.windowsdeveloper.de

Fahrplan fürs Web

Viele Wege führen zum Ziel

Sonderdruck für Ice Tea Group

Komplexe Geschäftsanwendungen fürs Web

Rapid Web Development

Nehme ich Angular oder doch lieber Blazor? Welche Technologie ist mittelfristig für browserbasierte Unternehmensanwendungen sinnvoll und am besten geeignet? Mit Rapid Web Development lassen sich die Vorzüge klassischer Entwicklungsansätze mit der Geschwindigkeit des Rapid Prototypings verbinden, um effizient webbasierte Lösungen in C# zu realisieren. Dank der Wisej-Technologie funktioniert das auch für große und komplexe Projekte.

von Thomas Althammer

Technologieentscheidungen sind alles andere als einfach. Bei der Neuentwicklung von Geschäftsanwendungen steht der Browser als Ziel heutzutage (fast) immer fest. Neben einem unüberschaubaren Angebot an möglichen Technologien und Programmiersprachen konkurrieren neue Ansätze um die Gunst der Entwicklerinnen und Entwickler. Sie müssen sich nicht mehr nur zwischen C#, VB.NET, PHP oder JavaScript entscheiden. Auch No-Code bzw. Low-Code oder Codegeneratoren verheißen neue Möglichkeiten. Doch spiegelt das die echte Welt und den realen Bedarf wider?

Bei der Realisierung von Unternehmensanwendungen ist mehr gefragt als eine gute Marketingbotschaft. Das Entwicklerteam wird viele Jahre mit dem Technologiestack arbeiten; der entstehende Code muss sich gut strukturieren, warten und leicht weiterentwickeln lassen. Die durchschnittliche Lebensdauer von komplexeren Geschäftslösungen kann schnell 20 und mehr Jahre betragen.

Gesucht: einfach, skalierbar, leistungsfähig

Aufgrund der zunehmenden Digitalisierung werden sich Anforderungen an Softwarelösungen im geschäftlichen Umfeld in den nächsten Jahren stark weiterentwickeln. Zugleich stehen immer weniger Entwickler zur Verfügung, um effizient und mit ordentlicher Produktivität die Erwartungen der Anwenderinnen und Anwender zu erfüllen. Für die Basistechnologie des neuen Entwicklungsprojekts wird also die eierlegende Wollmilchsau gesucht, um den folgenden Kriterien möglichst nahe zu kommen:

- möglichst geringe Anzahl an Sprachen/Technologien über den ganzen Stack hinweg
- flache Lernkurve und geringer Programmieraufwand auch für komplexe Anforderungen
- hohe Wartbarkeit über Jahre hinweg
- große Flexibilität, um auch zukünftigen Anforderungen zu begegnen
- Plattformunabhängigkeit, z. B. für verschiedene Varianten und Endgeräte
- ordentliche Produktivität, mit What-you-see-is-what-you-get-(WYSIWYG-) Designer und Vererbbarkeit visueller Klassen
- gute Skalierbarkeit, auch in umfangreichen Anwendungsfällen

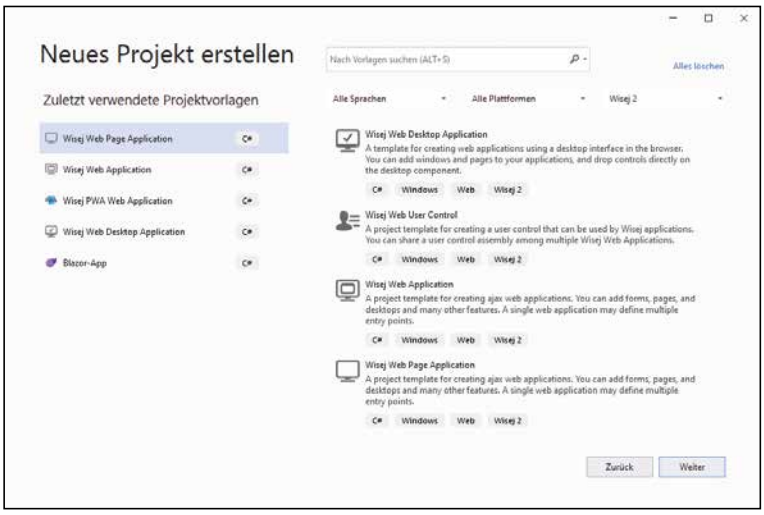

Abb. 1: Neue Projektvorlagen in Visual Studio nach der Installation von Wisej

Für das Rapid Web Development stehen verschiedene Angebote bereit. Im .NET-Umfeld hat sich in den letzten Jahren das Wisej Framework hervorgetan, das mittlerweile auch bei größeren Unternehmen wie DATEV, Goodyear oder der HARTMANN GRUPPE zum Einsatz kommt. Dabei gibt es drei wesentliche Besonderheiten im Vergleich zu anderen Toolkits und UI Frameworks:

1. Die Entwicklung der Oberfläche erfolgt konsequent nach dem WYSIWYG-Prinzip im Visual Studio Designer. Die Benutzeroberfläche wird auf den Pixel genau schon zur Entwurfszeit bearbeitet.
2. Entwickelt wird ausschließlich in C# beziehungsweise einem .NET-Dialekt nach Wahl. Der Einsatz von HTML, CSS oder JavaScript ist auch in komplexen Szenarien nicht erforderlich. Wisej kann aber um externe JS-Bibliotheken umfangreich erweitert werden.
3. Wisej stellt die Oberfläche als native HTML5 Single Page Application (SPA) in allen gängigen Browsern dar. Client und Server werden dabei stets über ein leichtgewichtiges Echtzeitprotokoll synchron gehalten. Mittels Theming lässt sich die Oberfläche umfangreich anpassen und gestalten.

Rapid Web Development im Einsatz

Bevor die Entwicklung mit Wisej beginnen kann, ist zunächst der Download einer kostenlosen Testversion erforderlich. Eine solche Trial-Version ist auf der Website des Herstellers Ice Tea Group erhältlich [1]. Wem die 15 Tage als Testzeitraum nicht genügen, kann sich die Evaluationslizenz auf Anfrage verlängern lassen. Für das Jahr 2021 ist eine Community Edition von Wisej angekündigt.

Nach der Installation finden sich einige neue Symbole auf dem Desktop. Doch die eigentliche Arbeit mit Wisej beginnt in Visual Studio. Im Dialog Neues Projekt erstellen fällt eine Reihe von Projektvorlagen ins Auge, bei der wir uns für die Wisej Web Page Application entscheiden (**Abb. 1**). Der wesentliche Unterschied der Projektvorlagen besteht in der Darstellung der Oberfläche der Webanwendung. Wisej kann hier neben der üblich vollflächigen Webseitendarstellung auch fensterbasierte Anwendungen oder sogar eine desktopartige Oberfläche für verschiedene Apps im Browser realisieren.

Migration ins Web als Alternative zur Neuentwicklung?

Es muss nicht immer eine Neuentwicklung sein. Mit Hilfe des „Web-Enabling“ lassen sich vorhandene Desktopanwendungen innerhalb weniger Monate in Single Page Applications für den Browser übertragen. Was auf den ersten Blick unmöglich erscheint, hat den Praxistest einer Vielzahl von Projekten weltweit bestanden.

Die in diesem Beitrag vorgestellte Wisej-Technologie erlaubt als Alternative zu Rapid Web Development die Migration auch komplexer und gewachsener Codestrukturen ins Web. Die eigentliche Geschäftslogik muss dabei häufig nicht oder nur sehr eingeschränkt modifiziert werden. Selbst eine Vermischung von Oberflächencode mit Geschäftslogik kann meist ohne Änderungen übernommen werden und läuft 1:1 als native HTML5-Anwendung weiter. Die Vorteile des Web-Enabling liegen auf der Hand:

- Schnelle Umstellung vom Desktop in den Browser
- Vorhandene Applikationslogik kann weiter genutzt werden
- Erhebliche Zeit- und Kosteneinsparung
- Flache Lernkurve und sehr einfacher Umstieg für das Team
- UI-Redesign, Refactoring und Erweiterung uneingeschränkt möglich

Die Migration von Bestandsanwendungen wird für folgende Programmiersprachen angeboten: Windows-Forms-Lösungen (C#, VB.NET), Visual Basic 6, Visual WebGUI, Gupta/Centura Team Developer, Microsoft Access. Eine teilautomatisierte Migration lässt sich realisieren für COBOL, Xbase++, C++ und andere Sprachen.

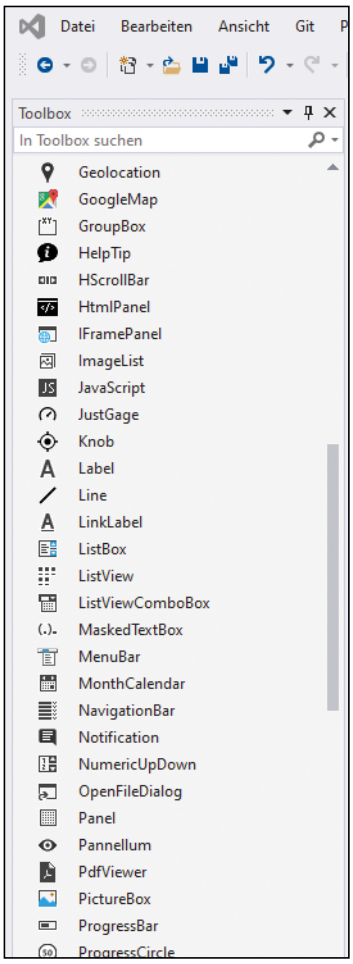

Abb. 2: Wisej enthält eine Vielzahl von Steuerelementen

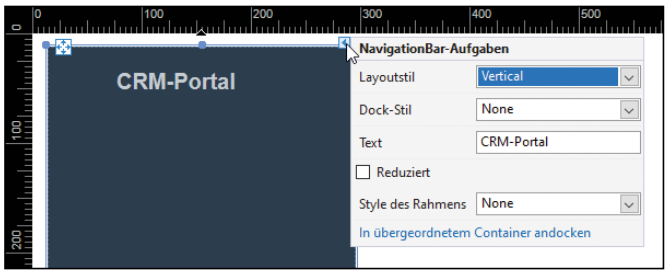

Abb. 3: Aufgabenfenster in Wisej

Ein CRM-Prototyp im Handumdrehen

Nach Bestätigung des Speicherorts öffnet sich direkt der Wisej Designer für Page1 im neuen Projekt – benennen Sie diese am besten in MainPage um. In der Toolbox sehen Sie eine Vielzahl von Controls, die Wisej bereits im Auslieferungszustand mitbringt (**Abb. 2**), es kann außerdem um beliebige JavaScript Controls von Drittherstellern oder aus dem Open-Source-Umfeld erweitert werden. Wählen Sie die NavigationBar und fügen Sie eine Instanz auf der leeren MainPage hinzu. Es öffnet sich direkt ein kleines Aufgabenfenster, das die wichtigsten Eigenschaften im Designer einstellbar macht. Tragen Sie einen passenden Titel und als Docking Left ein, damit die NavigationBar vollflächig erscheint (**Abb. 3**). Wesentliche Eigenschaften von Wisej Controls können direkt im Designer mit Hilfe des kleinen Pfeils über den Aufgabendialog bearbeitet werden. Die Items-Eigenschaft der NavigationBar wird nun mit Einträgen versorgt. Legen Sie zwei Menüeinträge an, z. B. Kunden (mit Tag KD) und Mitarbeitende (mit Tag MA).

Über das öffentliche GitHub-Repository [2] stehen für Wisej zahlreiche Erweiterungen im Quellcode zur Verfügung. Gleich mehrere Extensions enthalten Iconbibliotheken, die im SVG-Format zur Nutzung bereitstehen. Nach Kompilieren der Assembly können sie einfach als Verweis hinzugefügt und über den integrierten Dialog als Icons für NavigationBar-Items ausgewählt werden. Wisej erlaubt das Einfärben der Icons mit Farben direkt aus der Dialogoberfläche heraus. All das kann wahlweise auch mittels Programmcode gemacht werden, z. B., wenn Einträge für die Oberfläche aus der Datenbank geladen werden sollen. Bei statischen Einträgen ist das im Designer schnell gemacht und zur Entwurfszeit praktisch, da wir das Look and Feel der Anwendung sofort kontrollieren können (**Abb. 4**).

Sämtliche Einstellungen zur Entwurfszeit werden vom Wisej Designer in sogenannten *designer.cs*-Dateien gespeichert, ganz ähnlich wie das aus dem Windows-Forms-Umfeld her bekannt ist (Listing 1). Eine Wisej-

Listing 1

```
#region Wisej Designer generated code

/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
  this.navigationBar1 = new Wisej.Web.Ext.NavigationBar.NavigationBar();
  this.navBarItemKD = new Wisej.Web.Ext.NavigationBar.NavigationBarItem();
  this.navBarItemMA = new Wisej.Web.Ext.NavigationBar.NavigationBarItem();
  this.panel1 = new Wisej.Web.Panel();
  this.SuspendLayout();
  //
  // navigationBar1
  //
  this.navigationBar1.Dock = Wisej.Web.DockStyle.Left;
  this.navigationBar1.Items.AddRange(
                    new Wisej.Web.Ext.NavigationBar.NavigationBarItem[] {
  this.navBarItemMA,
  this.navBarItemKD});
  this.navigationBar1.Name = "navigationBar1";
  this.navigationBar1.Size = new System.Drawing.Size(258, 700);
  this.navigationBar1.TabIndex = 0;
  this.navigationBar1.Text = "CRM-Portal";
  this.navigationBar1.ItemClick +=
          new Wisej.Web.Ext.NavigationBar.NavigationBarItemClickEventHandler(
                                        this.navigationBar1_ItemClick);
  //
  // navBarItemKD
  //
  this.navBarItemKD.BackColor = System.Drawing.Color.Transparent;
  this.navBarItemKD.Icon = "resource.wx/Wisej.Ext.MaterialDesign/
                                briefcase-with-tick-inside.svg?color=#00FF9F";
  this.navBarItemKD.Name = "navBarItemKD";
  this.navBarItemKD.Tag = "KD";
  this.navBarItemKD.Text = "Kunden";

  ...
}
#endregion
...
```

Anwendung ließe sich auch gänzlich ohne Designer erstellen.

Anzeige von Modulen auf der rechten Seite vorbereiten

In unserem CRM-Prototyp sollen auf der rechten Seite, also neben der NavigationBar, nun die Module geladen werden. Das kann wahlweise direkt erfolgen, wir entscheiden uns jedoch für ein Panel als Container, um im Header noch eine passende Überschrift zu unserem Modul einzublenden. Die Schritte sind ähnlich einfach wie bei der NavigationBar umgesetzt:

1. Panel aus der Toolbox wählen und im freien Bereich unserer MainPage als Instanz erstellen.
2. Mit Hilfe des kleinen Aufgabenpfeils oben rechts an der Control oder alternativ über die Eigenschaften wird der Header aktiviert (ShowHeader = True).
3. Einen Schließen-Button benötigen wir an dieser Stelle nicht (ShowCloseButton = False), das Docking setzen wir auf Fill und wählen als Theme-Farbe für die HeaderBackColor nun NavbarBackground, um hier ein einheitliches Look and Feel zu erzeugen.

Wisej erlaubt die Anpassung von Aussehen und Farben anhand von Themes, sodass mit wenig Aufwand verschiedene Farbschemata erstellt werden können.

Module erstellen und laden

Der grobe Rahmen steht. Nun geht es daran, die Anwendung mit Funktion und Inhalt zu füllen. Von den vielen Wegen und Möglichkeiten, das zu realisieren, entscheiden wir uns für das Laden von UserControls, die auf der rechten Seite im Panel erscheinen. Mit der rechten Maustaste auf unserem Projekt fügen wir im Projektmappen-Explorer von Visual Studio ein Neues Element hinzu und wählen im Bereich Wisej 2 die entsprechende Basisklasse aus (**Abb. 5**). Wisej liefert eine Vielzahl von Basisklassen mit, und es können eigene Klassen erstellt bzw. ergänzt werden (inkl. Mehrfachvererbung für visuelle Controls). Bei der Gelegenheit geben wir gleich die Bezeichnung *ModulKunden.cs* bzw. *ModulMitarbeitende.cs* an. Wer mag, kann für die UserControl selbst der Optik wegen noch ein Padding setzen, bevor wir die Controls ergänzen.

Das Öffnen der Module übernehmen wenige Zeilen Code, die nun dafür sorgen, dass Kunden und Mitarbeitende an der richtigen Stelle angezeigt werden. Ausschlaggebend ist in MainPage das Ereignis ItemClick der NavigationBar, für das der Code aus Listing 2 ergänzt werden muss.

Data Binding in Aktion

Im neuen Modul Kunden soll eine Liste der Ansprechpartner aus der guten alten Northwind-Datenbank mittels Data Binding angezeigt werden. Vielleicht haben

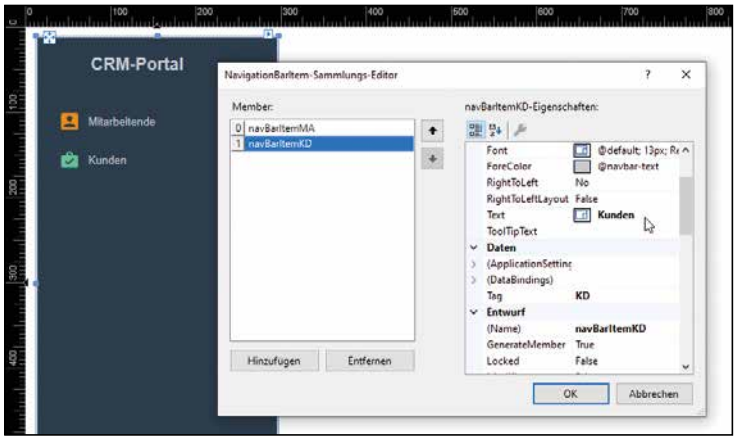

Abb. 4: Einträge der NavigationBar lassen sich im Entwurfsmodus mit Hilfe eines Editors anlegen

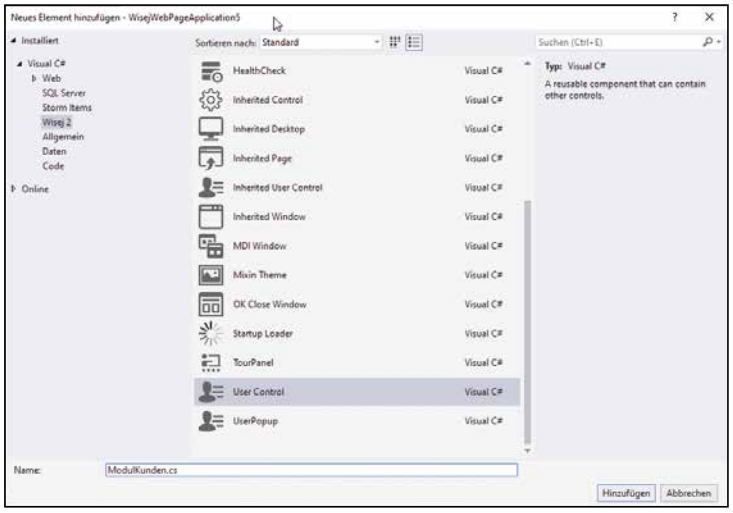

Abb. 5: Wisej liefert eine Vielzahl von Basisklassen mit

Listing 2

```
private void navigationBar1_ItemClick(object sender,
        Wisej.Web.Ext.NavigationBar.NavigationBarItemClickEventArgs e)
{
  Control currentModule = null;
  switch (e.Item.Tag)
  {
    case "KD":
      currentModule = new ModulKunden();
      break;
    case "MA":
      currentModule = new ModulMitarbeitende();
      break;
  }

  this.panel1.Controls.Clear();
  if (currentModule != null)
  {
    this.panel1.Text = e.Item.Text;
    this.panel1.Controls.Add(currentModule);
    currentModule.Dock = DockStyle.Fill;
  }
}
```

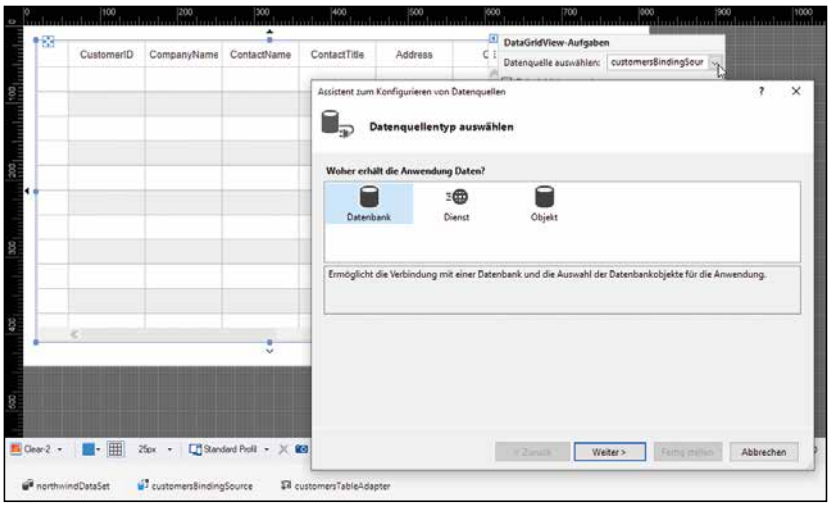

Abb. 6: Für die DataGridView werden die Daten mittels Data Binding geladen

Sie eine andere Beispieldatenbank zur Hand, die Sie hier als Datenquelle anbinden können. Unter [3] steht Ihnen eine Version im Access-Format für Testzwecke zur Verfügung. Auf Basis des .NET Frameworks stehen verschiedene Möglichkeiten bereit, um Daten abzufragen, zu manipulieren und in Benutzeroberflächen anzuzeigen. All diese Möglichkeiten werden von Wisej unterstützt, solange ein entsprechender Adapter bzw. Treiber für die Datenquelle verfügbar ist. In diesem Beitrag entscheiden wir uns für ein direktes Data Binding ohne Umweg über das Entity Framework oder andere objektrelationale Mapper.

Auf unsere neu erstellte UserControl für Kunden fügen wir eine DataGridView (DGV) hinzu. Das Docking der DGV wird auf Fill gestellt. Über das kleine Aufgabenfenster stellen wir nun die Datenquelle ein und wählen Projektdatenquelle hinzufügen, geben die Microsoft-Access-Datenbank an (wobei jede andere Art von Datenquelle natürlich entsprechend möglich wäre) und überlassen Wisej die restliche Arbeit. Zwischendurch haben wir uns für die Tabelle Customers entschieden. Nach Abschluss des Assistenten finden sich eine customersBindingSource und ein customersTableAdapter in unserer UserControl, und in der DataGridView erscheinen die Spalten aus der Datenbanktabelle, wie in **Abbildung 6** dargestellt. Wisej fügt automatisch die nötige Datenquelle und den TableAdapter hinzu. Wir können nun die Spalten bearbeiten und z. B. die CustomerID ausblenden und für andere Spalten mit der Eigenschaft „AutoEllipsis“ dafür sorgen, dass bei zu wenig Platz der Inhalt automatisch mit „…“ endet.

Ein klein wenig Code ist an dieser Stelle aber doch zu ergänzen: Mittels Doppelklick auf UserControl landen wir im Code-Editor und fügen dem Load Event folgende Zeilen hinzu:

```
private void ModulKunden_Load(object sender, EventArgs e)
{
  this.customersTableAdapter.Fill(this.northwindDataSet.Customers);
  this.dataGridView1.AutoResizeColumns();
}
```

Mit Hilfe der Toolbar oder *F5* wird das Debugging gestartet. Der Browser öffnet sich und die Wisej-Anwendung erscheint. Beim Klick auf Kunden in der NavigationBar sollte rechts die entsprechende Liste im Grid erscheinen.

Im Browser läuft nun das von uns erstellte Programm als HTML5-Anwendung mit CSS und JavaScript. Wir mussten bisher keine einzige Zeile JavaScript-Code schreiben, um dieses Ergebnis zu realisieren. Auch im weiteren Verlauf kommen wir gänzlich ohne JavaScript aus und bleiben ausschließlich bei C#, selbst in äußerst komplexen Anwendungsszenarien. Alle lästigen Details wie Session-Management, Browser-Server-Kommunikation und das Mapping zwischen clientseitigem JavaScript und dem serverseitigen .NET-Code übernimmt Wisej für uns.

Umfangreiche Debugging- und Erweiterungsmöglichkeiten

Obwohl Wisej bereits viele Möglichkeiten und Optionen enthält, besteht manchmal doch der Wunsch, auch eine externe JavaScript Control einzusetzen. In so einem Moment kommt man für die Integration nicht ganz ohne JavaScript aus, um die Control in Wisej zu verpacken. Die Mühe lohnt sich: Es ist dann mit WYSIWYG schon beim Design der Oberfläche in Visual Studio zur Entwurfszeit so dargestellt wie später zur Laufzeit. Für Wisej stehen neben den bereits erwähnten Extensions bereits fertige Implementierungen für die JavaScript-Bibliotheken der bekannten Hersteller DevExpress, Syncfusion, Telerik Kendo UI, ComponentOne und Infragistics fertig zur Nutzung bereit [4]. Wir haben damit Zugriff auf Hunderte teils sehr komplexe Controls für alle möglichen Anwendungsfälle. Wenn das nicht genügt, bietet die Wisej-Homepage über die Blogserie „All about integration“ [5] eine mehrteilige Anleitung, wie JavaScript Controls selbst in Wisej eingebunden werden können.

Interessant ist in diesem Zusammenhang das Debugging: Wir können wie gewohnt an beliebigen Stellen im Code Breakpoints setzen und das Verhalten der Anwendung untersuchen, wenn einmal Probleme auftreten sollten. Variablen lassen sich so anzeigen bzw. verändern, und gerade in komplexeren Settings behalten wir stets die Kontrolle über den Programmablauf, das Verhalten und die Darstellung der Anwendung. Im Vergleich zur „normalen“ Webentwicklung mag das ungewohnt erscheinen. Auf Basis von Wisej wurden bereits komplexeste Geschäftsanwendungen erstellt, die teils über Hunderte von Formularen/Seiten und mehrere Millionen Zeilen Code verfügen. Da ist es für Entwickler äußerst hilfreich, schon im Entwurfsmodus genau zu sehen, wie sich die Anwendung später zur Laufzeit verhält, um gezielt eingreifen und umfangreich debuggen

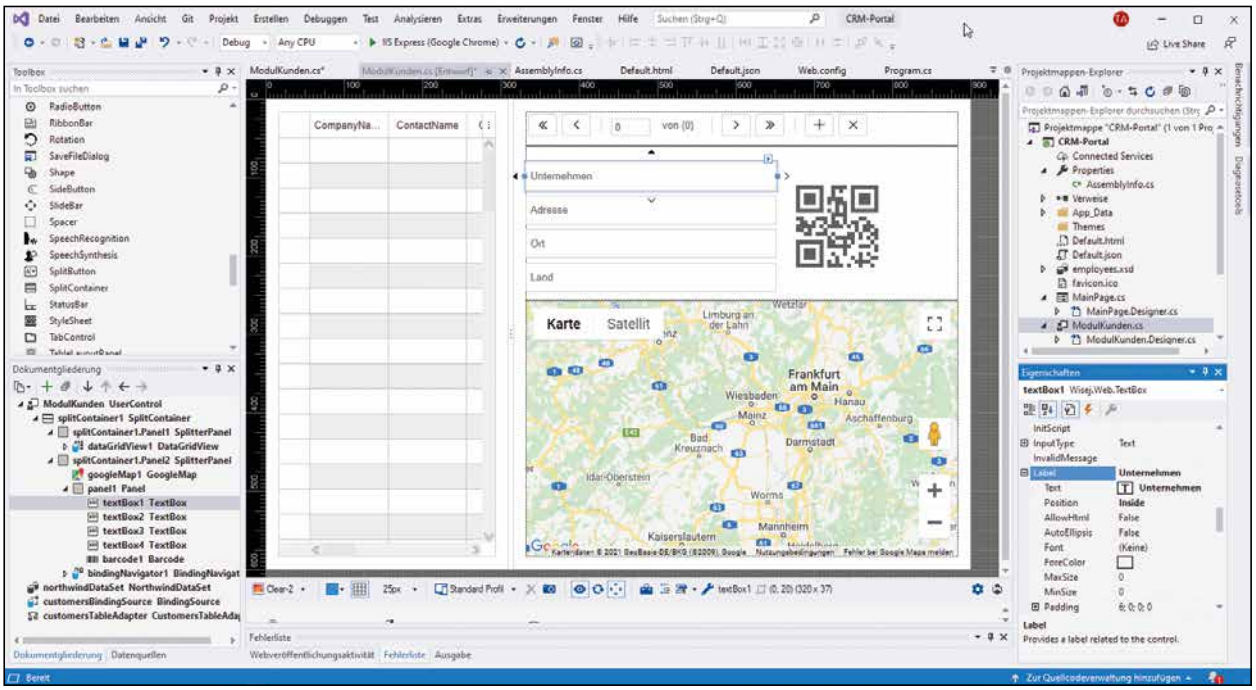

Abb. 7: Die Kundenverwaltung

zu können.

Erweiterung mit Detailansicht, QR-Code und Google-Maps-Integration

Basierend auf dem oben vorgestellten Rahmen kann nun die Anwendung vielfältig erweitert werden. Ihrer Kreativität sind dabei keine Grenzen gesetzt. Der in diesem Artikel vorgestellte CRM-Prototyp ist als Beispielanwendung im Quellcode verfügbar und um einige weitere Funktionen ergänzt worden. Folgende Elemente und Funktionen sind dabei beispielsweise einen näheren Blick wert:

- Erweiterung der Kundenansicht um eine Master-Detail-Anzeige: Neben der DataGridView wurde ein BindingNavigator ergänzt, der eine datensatzbasierte Navigation ermöglicht. Die Auswahl wird automatisch in der DGV markiert, und Datenfelder zeigen die Inhalte im Detail an, nachdem sie in das Data-Binding mit einbezogen wurden. Es handelt sich um normale Felder vom Typ *TextBox*.
- Mittels QR-Code soll die Adresse eines Kunden abfotografiert werden können. Die Barcode-Control wurde ergänzt und auf den Barcode Type *QR* gesetzt.
- Direkt in der Datensatzansicht soll die Möglichkeit bestehen, die Adresse in einer interaktiven Karte darzustellen. Hierfür wurde eine Google Map Control eingefügt und die Standardkoordinaten wurden voreingestellt (kann über die Eigenschaften oder direkt im Programmcode erfolgen). Bevor das lauffähig ist, tragen Sie bitte in den Eigenschaften der Control einen persönlichen API-Key ein, den Sie für Testzwecke direkt bei Google kostenfrei erhalten.

Die vorgestellten Erweiterungen wurden fast ohne manuelles Coden realisiert. Neben den zwei oben erwähnten Zeilen zum Laden der Kundenliste ist nach Ergänzung der Control lediglich der C#-Code aus Listing 4 einzufügen.

Bei Durchsicht des zum Download beigefügten Source Codes [6] lohnt ein Blick auf die Dokumentengliederung in Visual Studio. Hier helfen ein SplitContainer, zwei SplitContainerPanels sowie ein weiteres Panel, um die Anordnung und automatische Größenanpassung ergonomisch zu gestalten. **Abbildung 7** zeigt das fertige Ergebnis: Die Kundenverwaltung als Prototyp mit einigen Feldern für die Detailanzeige, einem QR-Code und Google-Maps-Integration kommt mit wenigen Zeilen C#-Code daher.

Darstellung von Mitarbeitenden mittels DataRepeater

Als zweites Modul für unseren Prototyp soll eine Darstellung von Mitarbeitenden aus der nostalgischen Northwind-Datenbank entstehen. Freuen Sie sich also

Listing 4

```
private void customersBindingSource_CurrentChanged(object sender, EventArgs e)
{
  string address = textBox2.Text + ", " + textBox3.Text + ", " + textBox4.Text;
  googleMap1.ClearMarkers();
  if (address.Length > 10)
  {
    googleMap1.AddMarker(textBox1.Text, address);
    googleMap1.CenterMap(address);
    barcode1.Text = address;
  }
}
```

auf Mitarbeiterfotos aus den frühen 90er Jahren. Anstelle einer Grid- und feldbezogenen Darstellung wie bei der Kundenverwaltung soll hier lieber die *DataRepeater* Control genutzt werden. Im Gegensatz zum Grid ist dabei die Darstellung von Zeilen flexibel gestaltbar: Wir definieren Felder und Anordnung je Datensatz in einem ItemTemplate, das dann für jeden Mitarbeitenden ausgegeben wird. Werfen wir am besten einmal einen Blick auf *ModulMitarbeitende.cs* im beigefügten Quellcode [6]. Die in den *TextBox* Controls enthaltenen Labels werden hier genutzt, und wieder eine BindingSource, um den *DataRepeater* zu befüllen.

Wisej kann in allen datengebundenen Controls auch mit sehr großen Datenmengen umgehen. Bei Bedarf sorgt das Virtual Scrolling dafür, dass stets nur der sichtbare Teil einer Control an den Browser übertragen wird. Im Vergleich zum vollständigen Laden und Übertragen eines *DataSet* ergeben sich hierüber erhebliche Geschwindigkeitsvorteile. Als Entwickler muss ich mich um das ganze Handling so gut wie nicht kümmern, da es sich um Standardfunktionen in Wisej handelt, die automatisch sinnvoll vorbelegt sind. Bei Bedarf kann manuell mit Feintuning eingegriffen werden. Erwähnt sei hier exemplarisch, dass die Anzahl nachzuladender Datensätze für ein möglichst ruckelfreies Scrollen definierbar ist. Gerade bei der Touch-Bedienung auf Tablets und Smartphones ist das eine tolle Sache, weil trotz riesiger Datenmengen und virtuellem Laden der Daten das Scrolling so flüssig erscheint wie bei vollständig geladenen Datenquellen.

Zusammenfassung

Das Framework Wisej entpuppt sich bei näherer Betrachtung als eierlegende Wollmilchsau. So können auf der einen Seite Prototypen und ganze Anwendungen mit wenig Aufwand effizient für den Browser entwickelt werden. Der Hersteller stellt zugleich auf seiner Homepage eine Reihe von Projekten vor, die das Skalierungspotenzial von Wisej aufzeigen: Namhafte Unternehmen setzen das Werkzeug auch in äußerst komplexen Szenarien für umfangreiche Geschäftsanwendungen ein. Der in diesem Artikel vorgestellte Rapid-Web-Development-Ansatz soll nur eine Perspektive darstellen und rückt in komplexen Projekten häufig etwas in den Hintergrund. Stattdessen werden bei größeren Anwendungen Klassenbibliotheken mit Mehrfachvererbung genutzt, um ganze ERP-Lösungen in den Browser zu verlagern. Der Hersteller Ice Tea Group bietet zusammen mit Partnerunternehmen an, bestehende Windows-Desktop-Anwendungen auf diese Weise in webbasierte Programme zu verwandeln, ohne dass hierfür bestehende Anwendungslogik in größerem Stil angepasst werden müsste.

Der Rapid-Development-Ansatz soll nur eine Perspektive darstellen und rückt bei komplexen Projekten häufig etwas in den Hintergrund.

Für das Jahr 2021 soll mit Wisej 3 der Umstieg auf .NET Core erreicht werden. Damit können Wisej-Anwendungen dann auch serverseitig unter Linux oder dem Mac bereitgestellt werden. Mit Wisej Mobile hat der Hersteller als Teil der Version 2.2 vor wenigen Monaten bereits Unterstützung für mobile Geräte auf den Markt gebracht.

Thomas Althammer verbindet Softwareentwicklung mit den Themen Datenschutz und Informationssicherheit. Er ist Mitbegründer der Ice Tea Group, LLC mit Sitz in Washington, DC und Geschäftsführer des Datenschutzdienstleisters Althammer & Kill GmbH & Co. KG.

.NET 6 kommt ohne Designer für WinUI 3.0

Nach der Veröffentlichung der zweiten Preview von .NET 6 hat Microsoft bestätigt, dass die für November 2021 geplante Freigabe von WinUI 3.0 ohne einen grafischen XAML Designer erscheinen wird [7].
Nach den Irrungen und Wirrungen rund um die Ansätze UWP, MAUI, WinUI und Project Reunion ist die Verunsicherung weiterhin groß, welchen Weg Microsoft für die Entwicklung von Programmoberflächen beschreiten möchte. Eine langfristige Strategie mit Planungssicherheit und Rückwärtskompatibilität ist aktuell nicht eindeutig erkennbar.

Links & Literatur

[1] http://www.wisej.com

[2] https://github.com/iceteagroup/wisej-extensions

[3] Northwind-Datenbank im Access-Format: https://drive.google.com/file/d/1-cOnpaZwfYiNIUDbiInjKu4qXlotAG68/view

[4] Details zu den Premium Extensions in Wisej 2.2 (verfügbar über das öffentliche GitHub-Repository): https://wisej.com/wisej-2.2/

[5] Tutorials zur Integration weiterer JavaScript Controls in Wisej: https://wisej.com/tag/integration/

[6] Der vollständige Quellcode aus diesem Artikel in C# mit Wisej: https://drive.google.com/file/d/18zLydP7KfGFfaFjvNcm2u-lu6mKeNnTW/view

[7] https://devblogs.microsoft.com/dotnet/announcing-net-6-preview-2/#comment-8727