Moderne Anwendungsarchitekturen

API-Services sind die Grundlage moderner verteilter Applikationen.

Microservices gelten heute als Basis moderner Anwendungsarchitekturen. Statt einer monolithischen und schwerfälligen Software wird die gesamte Funktionalität auf überschaubare Teilsysteme verteilt. Die entstandenen Services werden dann üblicherweise über das Internet und RESTful APIs zur Verfügung gestellt. Mit RAD Studio und dem RAD Server kann man vom Client bis zum Backend mit einer technologischen Basis arbeiten. Das ist effizient und bedeutet Wettbewerbsvorteile durch eine Verringerung der Time-To-Market.

Die Art und Weise der Softwareentwicklung steht einmal mehr am Scheideweg. Es scheint so, als ob die Zeiten komplexer und teilweise schwerfälliger Softwareapplikationen aus der Mode gekommen sind. Stattessen hat man es überall mit einer hohen Vielfalt und Flexibilität zu tun. Neben unterschiedlichsten Gerätetypen kommen auch die verschiedensten Betriebssysteme zum Einsatz. Auf dem Desktop sind es die Betriebssysteme Windows, macOS und Linux und für die mobilen Geräte iOS und Android. Ebenso vielfältig ist die Geräteauswahl. Neben klassischen Desktop-PCs, werden Notebooks, Tablets und Smartphones in allen erdenklichen Leistungsklassen und Bauarten verwendet. Für die Softwareentwicklung haben diese Entwicklungen vielfältige Konsequenzen. Insbesondere ändert sich die verwendete Architektur der Anwendungssysteme. Der Weg geht vom monolithischen Anwendungssystem, hin zu einer schlanken und flexiblen Architektur auf der Basis von Microservices. Ebenso gilt es bestehende und bewährte Applikationen im Zuge der Migration für diese Ansprüche fit zu machen.

Mit den richtigen Werkzeugen, wie RAD Studio und RAD Server, gelingt es ohne Technologiebrüche, Software auf der Basis von Microservice effizient zu erstellen. Das ist die Voraussetzung für ein hohes Time-to-Market und die Generierung von Wettbewerbsvorteilen.

Nutzerperspektiven im Wandel

Statt einem stationären PC werden viele der alltäglichen Aufgaben heute auf mobilen Geräten durchgeführt. Nutzer möchten nicht in der Wahl Ihrer Technik und Plattform eingeschränkt werden. Hat man einen Flug am heimischen PC gebucht, dann möchte man unterwegs auf dem Smartphone auf das Kundenkonto zugreifen. Das Ticket zeigt man bequem als QR-Code bei der Kontrolle vor. Der Außendienstmitarbeiter einer Versicherung kalkuliert das kundenindividuelle Angebot direkt vor Ort; die Unterschrift erfolgt per Stifteingabe auf dem Tablet und die Dokumente können jederzeit über die Cloud vom Kunden eingesehen werden. Dazu genügt die Eingabe des Nutzernamens und eines Passwortes in einer App.

Das Gleichgewicht der Gerätenutzung verschiebt sich damit zunehmend hin zu einer Verwendung der mobilen Geräte (Abbildung 1).

Nutzer, gleichgültig ob im privaten oder betrieblichen Umfeld, möchten heute nach Möglichkeit die Software Ihrer Wahl von jedem beliebigen Ort aus, mit allen möglichen Geräten und unabhängig von der eingesetzten Systemumgebung nutzen. Es ist kaum noch vermittelbar, dass eine Anwendung an ein bestimmtes Betriebssystem gebunden ist oder die App nur für ein bestimmtes System bereitgestellt wird.

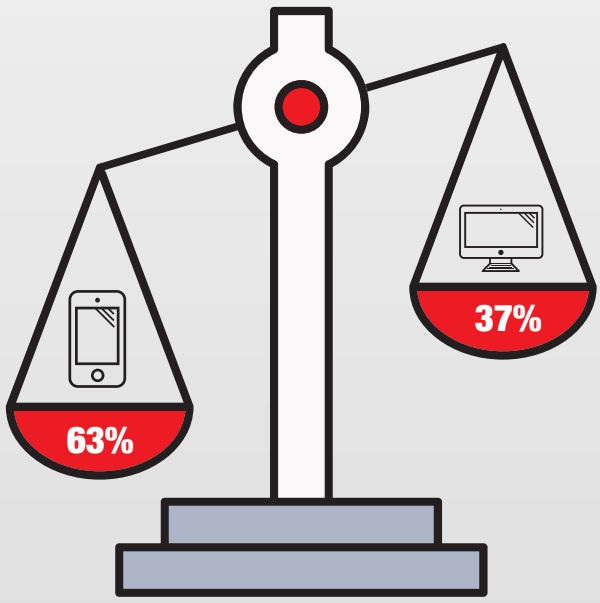

Abbildung 1 Die mobile Nutzung verdrängt den Desktop zunehmend

Als Entwickler sind wir daher gefordert, diese Ansprüche durch moderne und hoch dynamische Software abzudecken. Ein Weg dorthin ist die geräte- und plattformübergreifende Software auf Clientseite und eine flexible Gesamtsystemarchitektur, welche die Funktionalität als lose miteinander gekoppelte Services bereitstellt.

Technische Treiber

Es gibt eine Reihe von technischen Treibern, welche die Welt der Softwareentwicklung heute in einem besonderen Maße prägen. Eine dieser Entwicklungen ist die permanente Verfügbarkeit von schnellem Internet. Man kann davon ausgehen, dass eine Applikation nahezu einen stetigen Zugriff über das Internet auf Daten und Services hat. Insbesondere die Verfügbarkeit von schnellen Internetverbindungen für mobile Geräte über LTE erlauben den Zugriff vom Smartphone und Tablet auf entfernte Services und größere Datenmengen. Oft wird noch die Volumenbegrenzung des mobilen Internets als Einschränkung empfunden. Die aktuelle Entwicklung zeigt aber auch eindeutig, dass die vertraglich bereitgestellten Datenvolumina nach und nach ausgeweitet werden. Neue Technologiestandards, wie kabelgebundenes Internet über Glasfaser oder 5G für das mobile Netz, werden zwar noch kontrovers bezüglich der zu erwartenden Kosten für die Bereitstellung diskutiert, aber langfristig werden sich diese Technologien durchsetzen. Zusammenfassend kann man als Entwickler davon ausgehen, dass die Nutzer künftig flächendeckend schnelles Internet in ausreichender Bandbreite und mit genügend Datenvolumen zur Verfügung haben werden. Natürlich sollte man programmtechnisch einen Ausfall oder entsprechende Versorgungslücken berücksichtigen.

Die stetige Verfügbarkeit von Internet erlaubt es, dass man vielfältige Funktionen über entfernte Services anbieten und nutzen kann. Dabei wird die Funktionalität auf viele kleinere Einheiten aufgeteilt, welche lose miteinander gekoppelt werden. Dieses Konzept erhöht die Flexibilität und als Programmierer kann man auf das Know-how von anderen Entwicklern und Serviceanbietern zurückgreifen, welche entsprechende Funktionen zur Verfügung stellen. Einmal programmierte Services können dann in mehreren Anwendungen auf einfachste Art und Weise wiederverwendet werden.

Auf diese Weise können auch Apps auf mobilen Geräten oder Web-Applikationen die Power einer umfassenden Rechnerlandschaft nutzen. Es können Daten und Funktionen verwendet werden, welche man als Entwickler einer einzelnen Anwendung nicht oder nur mit einem erheblichen Aufwand bereitstellen könnte. Ein gutes Beispiel ist die Interaktion mit KI-Diensten. Aus den Anwendungen werden Daten über definierte Schnittstellen an einen Service übertragen, dort werden sie verarbeitet und die Antwort wird an den anfragenden Client gesendet.

Zusammenfassung: Veränderungen im Nutzerverhalten und die technischen Treiber führen zu anderen Ansprüchen an die Gestaltung von Software und deren Architekturen.

Der Weg zur modernen Softwarearchitektur

Als monolithische IT-Systeme werden umfassende Softwareapplikationen bezeichnet, welche keine explizite Trennung in unterschiedliche Teilsysteme aufweisen. Die typischen Funktionen eines solchen Anwendungssystems, also Datenhaltung, Anwendungslogik und Benutzeroberfläche sind in diesem Fall eng gekoppelt und weitreichend zu einem komplexen Gesamtsystem miteinander verschmolzen.
Das gesamte Anwendungssystem stellt sich als ein großer und überschaubarer Block dar. Ein monolithisches IT-System führt zu folgenden Nachteilen:

- Hohe Komplexität: Das Softwaresystem ist umfassend und die Teilmodule weisen vielfältige gegenseitige Abhängigkeiten zueinander auf.
- Schlechte Wartbarkeit: Die Wartung und Fehlerbereinigung stellen sich sehr schwierig dar. Es gelingt nicht einzelne Teile der Anwendung gegen neuere Technologien auszutauschen, ohne dass dies direkte Einflüsse auf andere Bestandteile der Software hat. Anpassungen an einer Stelle im Quellcode lösen eine Kette von notwendigen Folgeanpassungen aus.
- Schwierige Wiederverwendung: Die direkte Wiederverwendung einzelner Softwarekomponenten ist ausgeschlossen.
- Geringe Flexibilität: Eine Erweiterbarkeit und eine Skalierbarkeit sind stark eingeschränkt.

Aus heutiger Sicht weisen monolithische IT-Systeme keine durchdachte Architektur mehr auf. Oft sind diese durch stetige Weiterentwicklung und Anpassung entstanden. Das ist heute nicht mehr Stand der Technik. Dennoch sind noch viele solcher Systeme im Einsatz. Sie werden als Legacy-Systeme bezeichnet. Ihre Modernisierung und Anpassung (Migration) ist schwierig.

Besser ist eine Aufteilung der Gesamtapplikation in Schichten, üblicherweise für die Datenhaltung, die Verarbeitung und die Präsentation. Bei Apps und Web-Applikationen ist der Einsatz einer solchen Anwendungsarchitektur Pflicht. Da die Systemressourcen hier beschränkt sind, müssen Funktionen, Services und Datenhaltung ausgelagert werden.

Im industriellen Umfeld haben sich die serviceorientierten Architekturen (SOA) als eine Form der verteilte Systemarchitektur etabliert. Das Prinzip von der SOA besteht darin, dass Nutzer (Service Consumer) komplexe Anwendung nahezu aus dem Zusammenfassen bzw. der Kombination von Service-Dienstleistungen realisieren können. Die dahinterstehende Idee ist, zusammengehörende Anwendungsfunktionen zu einer Einheit zu bündeln und diese als Dienstleistungen (SOA-Services) anzubieten. Auf diese Weise können die vorhandenen Services wiederverwendet und Entwicklungskosten gespart werden. Alle verfügbaren Dienste werden über eine Registry bekannt gemacht. Die tatsächliche Bindung zwischen den Softwareelementen Service Consumer und Service Provider erfolgt zur Ausführungszeit durch die dynamische Zuordnung. SOA zielt damit auf eine Flexibilisierung der Unternehmens-IT ab. Eine übergreifende Anwendung, d.h. die Nutzung von SOA-Services durch Dritte ist i.d.R. nicht vorgesehen. SOA wird gelegentlich auch als Vorläufer für Microservices angesehen. Dennoch ist davon auszugehen, dass beide Technologien nebeneinander bestehen werden.

Bei Microservices können die Module unabhängig voneinander in Produktion gebracht werden. Die Services können dabei flexibel durch unterschiedlichste Anwendungssysteme, welche nicht nur auf das eigene Unternehmen beschränkt sind, benutzt werden. Die Integration erfolgt üblicherweise über RESTful http, d.h. über eine einheitliche Schnittstelle. Die eigene Anwendung wird damit gewissermaßen zum Service, welcher sich wiederum aus der Nutzung anderer Dienste zusammensetzt. Für die Nutzung eines Service spielt dessen technische Umsetzung keine Rolle. Unterschiedliche Betriebssysteme und die Bereitstellung auf gänzlich unterschiedlichen Technologien sind grundsätzlich möglich. Einzelne Services können ausgetauscht oder unabhängig vom Gesamtsystem aktualisiert werden. Sofern man die Services von Drittanbietern nutzt, gibt man gewissermaßen die gesamte Verantwortung für den Betrieb und die weitere Entwicklung weiter. Nutzt eine App beispielsweise den Service eines KI- oder eines Bezahldienstes, so muss diese nur die Aufrufkonventionen des API kennen. Die technische Umsetzung des Dienstes ist nicht von Interesse. Die Weiterentwicklung des Dienstes steht vollständig in der Verantwortung des Anbieters. Sofern sich die Schnittstelle nicht ändert, haben Anpassungen am Dienst keine Auswirkungen auf die nutzende Software. Dienste können – aus welchen Gründen auch immer – getauscht werden.

Chancen und Herausforderungen von Microservices

Microservices bieten eine Reihe von Chancen:

- Jedes Modul kann einzeln konzipiert, implementiert, getestet und deployt werden.
- Die Module sind hinsichtlich der Technik untereinander unabhängig, zum Beispiel bei der Wahl der Programmiersprache.
- Fehlerkorrekturen und Updates an einem Modul beeinflussen andere Module nicht, sofern die Schnittstelle unverändert bleibt.
- Einzelne Module können sehr gut in unterschiedlichen Anwendungskontexten verwendet werden.
- Die Module sind technisch voneinander unabhängig.
- Auch eine fachliche Unabhängigkeit der Module kann untereinander erreicht werden
- Eine bessere organisatorische Zuordnung der Verantwortlichkeiten als bei einer monolithischen Softwarestruktur ist möglich.

Dennoch bedarf auch die Umsetzung eines Softwaresystems auf der Basis einer Microservice-Architektur einer fundierten Planung. Folgende Herausforderungen sind zu bewältigen:

- Entwurf einer umsetzungsfähigen Planung für die Zuordnung der der Services, zum Beispiel nach Schichten, wie Services auf Daten-, Logik und User Interface-Ebene.
- Einführung einer passenden Microservice-Infrastruktur, um beispielsweise das Deployment der einzelnen Services weitgehend zu automatisieren.
- Begrenzung und Auswahl der verwendeten Technologien auf ein akzeptables Maß.

Gerade der letzte Punkt kann zu einer echten Herausforderung werden. Beim Aufbau einer komplexen Softwarelandschaft sind bereits clientseitig durchaus die unterschiedlichsten Technogien zur Entwicklung der verschiedensten Anwendungstypen (Desktopanwendung, App für die mobilen Geräte, Web-App) notwendig. Sehr schnell erfordert die Umstellung auf eine Microservice-Architektur ein Beherrschen weitere zusätzlicher Technologien, wie Microservice-Framework, Docker, Kubernetes, serverseitiges JavaScript in zusätzlichen Frameworks wie Node.js und vieles mehr.

Gefragt sind daher Ansätze, welche für einen durchgängigen Entwicklungszyklus vom Client bis hin zum RESTful Service auf Seiten des Servers sorgen, um den gesamten Entwicklungsprozess effizient gestalten zu können.

Microservices mit RAD Studio und RAD Server

Ein solcher Ansatz kann die Kombination aus der integrierten Entwicklungsumgebung RAD Studio und dem RAD Server darstellen. Betrachtet man sich die Architektur eines Anwendungssystems auf der Basis dieser Komponenten, so ergibt sich die Struktur gemäß Abbildung 2.

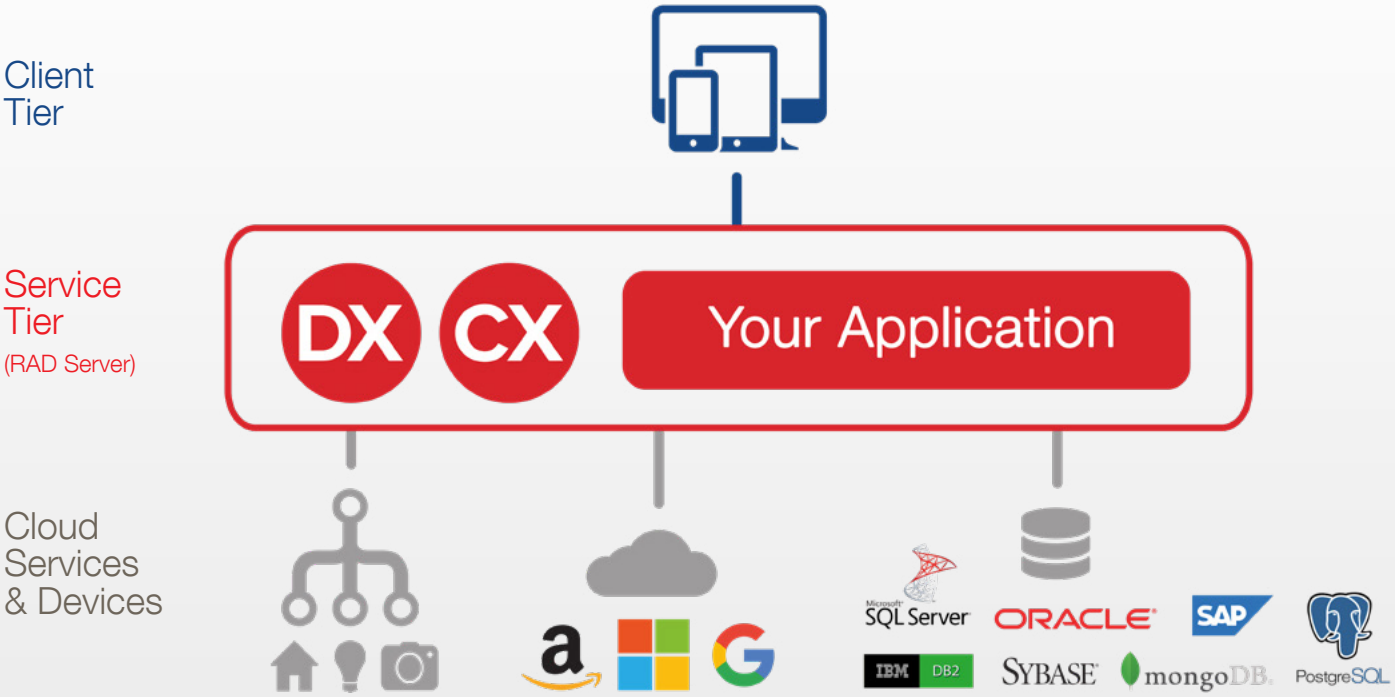

Abbildung 2 Architektur eines Anwendungssystems mit RAD Server als integrierende Middleware.

Die Server-Applikation fungiert hier flexibel und wahlfrei als Backend oder Middleware und bietet folgende Funktionen (Abbildung 3):

- Flexibel nutzbaren REST-Endpunkt: Clientseitig werden die Services als universelle REST/ JSON-API bereitgestellt und können damit nahezu von allen Arten von Clients (Desktop, Mobile, Web) genutzt werden. Eine integrierte Zugriffsteuerung erlaubt es Berechtigungen für die API-Nutzung über Benutzerauthentifizierung und -autorisierung festzulegen. Mit der integrierten API-Analyse bekommt man einen Überblick über die Nutzung der Services.
- Vermittelnde Zwischenschicht: Serverseitig bietet der RAD Server vielfältige Möglichkeiten, u.a. die Integration von Unternehmensdatenbanken, die Verwaltung von IoT-Hardware und die Integration von Cloud- und BaaS-Diensten, wie Google, Amazon, Facebook und Kinvey.
- Bereitstellung von Anwendungsdiensten: Der RAD Server bietet ein Set an Anwendungsdiensten wie Benutzerverzeichnis- und Authentifizierungsdienste, Push-Benachrichtigungen, Indoor-/ Outdoor-Positionsermittlung und JSON-Datenspeicher.

Abbildung 3 RAD Server als Serviceprovider und integrierende Middleware.

Erstellen von Clientoberflächen
Erstellen Sie in RAD Studio, Delphi oder C++ Builder auf schnelle Weise Ihre Oberflächenformulare für Desktopplattformen und Plattformen mobiler Geräte.

Verwalten Verwalten Sie Ihre APIs sowie Benutzer und analysieren Sie die Nutzung sowie die APIAktivität über das integrierte RAD Server-Portal zur Anwendungsverwaltung.

Bereitstellung Hosten Sie Ihren RAD Server auf einem privaten „lokalen" Windows-Server oder in einer Cloud von Amazon, Rackspace oder Azure.

Erstellen von Server-API-Endpunkten
Stellen Sie mit RAD Server Ihren serverseitigen Delphi- und C++-Programmcode bereit, wobei automatisch REST/JSON-API-Endpunkte für Ihre Clients erstellt werden.

Integrieren Verbinden Sie RAD Server mit Ihren Unternehmensdatenbanken, Cloud-Diensten und IoT-Geräten zur Erweiterung Ihrer Anwendung.

Hinzufügen von Anwendungsdiensten
Fügen Sie Benutzer und Regeln zur APIZugriffssteuerung hinzu und aktivieren Sie die Datenpersistenz zur Verwendung des integrierten sicheren Datastore.

Hinzufügen von Benutzern Konfigurieren Sie Gruppen und fügen Sie Benutzer über das RAD Server-Portal hinzu oder importieren Sie diese über LDAP. Weisen Sie Rechte für den API-Zugriff zur Festlegung der Funktionalität zu, auf die unterschiedliche Gruppen Zugriff haben sollen.

Ein besonderes Leistungsmerkmal des RAD Servers ist die nahtlose Bereitstellung der Dienste über die integrierte Entwicklungsumgebung RAD Studio. Damit ist es auf einfachste Weise möglich Geschäftslogik als Restful API Service zu hosten. Aktuell werden der Microsoft IIS Server (Windows) und Apache Server (Windows, Linux) unterstützt. Für die Entwicklung gibt es einen „Development Server" mit integriertem Webserver.

Unterstützung der Anwendungsmigration

Software altert, d.h. Applikationen können zwar über viele Jahre im Einsatz sein, dennoch kommt irgendwann die Zeit für eine vollständige Erneuerung. Entweder man beginnt ganz von vorn oder man kommt auf dem Weg der Migration zum Ziel. Letzteres sollte man stets in Betracht ziehen, um Aufwand und Kosten in Grenzen zu halten.

Bei der Migration von bestehenden Softwareapplikationen geht es sehr oft darum, die ursprüngliche Gesamtapplikation, beispielsweise bestehend aus den typischen Schichten Datenhaltung, Businesslogik und Benutzerschnittstelle in kleinere Einheiten (Services) zur zerlegen und diese als voneinander losgelöste Dienste zum Beispiel via RESTful Services anzubieten. Um die Kosten der Migration in Grenzen zu halten, ist stets zu prüfen, ob bestehender Programmcode nach einer Überarbeitung weiterverwendet werden kann. Dieses ist bei der Nutzung des RAD Servers und RAD Studio möglich. Vorhandener Delphi- oder C++-Quellcode kann mit vertretbarem Aufwand zu einem universell nutzbaren Backend portiert werden. Der Aufwand für notwendige Migrationen von Legacy-Applikationen kann auf diese Weise merklich reduziert werden.

Durchgängig vom Client bis zum Server

Eine hohe Produktivität des gesamten Entwicklungszyklus entsteht, wenn man RAD Studio als zentrales Entwicklungswerkzeug verwendet. Auf der Basis einer komponentenbasierten Entwicklung können Applikationen für die folgenden Plattformen erstellt werden:

- Microsoft Windows: Anwendungen für den Desktop und den Server. Es entstehen native 32/ 64-Bit-Anwendungen. Diese haben keine Abhängigkeiten zu weiteren Bibliotheken. RAD Studio bietet die Option die fertige Anwendung zu einem APPX-Package zu packen und sie damit über den Microsoft Store zu deployen.
- macOS: Es werden die macOS Versionen ab OS X 10.11 Sierra unterstützt.
- Linux: Es lassen sich Server-Applikationen erstellen. Mit einem Zusatz können die auf FireMonkey basierten Anwendungen auch für Linux mit einem grafischen UI ausgestattet werden.
- iOS: Apps für iPhone und iPad ab den Versionen iOS 10.
- Android: Apps für Smartphones und Tablets.

Mit dieser breiten Palette an Betriebssystemen unterstützt RAD Studio nahezu alle relevanten Systeme und Gerätetypen auf der Clientseite. Serverseitig wird das Backend ebenso in RAD Studio erstellt und über den RAD Server auf einen Produktionsserver gehostet (Abbildung 4).

Die bereitgestellten Services können dank generischen RESTful API ohne Einschränkungen clientseitig durch alle erdenklichen Technologien und Softwaresysteme verwendet werden.

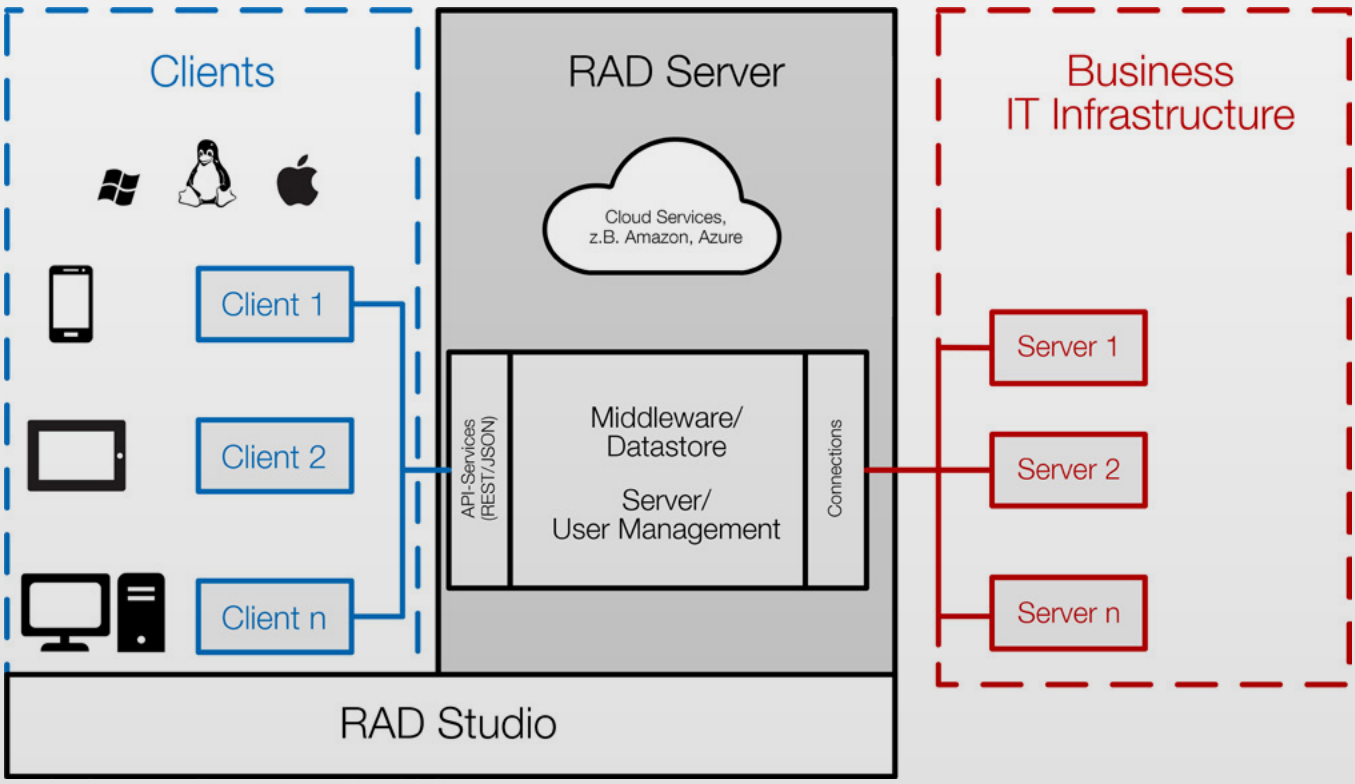

Abbildung 4 Durchgängig vom Client bis zum API Service mit RAD Studio.

Fazit

Die Architektur von Anwendungssystemen wird durch den Ansatz von Microservices flexibler. Die einzelnen Services werden voneinander entkoppelt. Damit gelingt es, die steigende Komplexität und Größe der Softwaresysteme in den Griff zu bekommen. Um den Aufwand, die Kosten und die Entwicklungszeit zu reduzieren, sollten nach Möglichkeit keine großen Systembrüche zwischen der Client- und Serverseitigen Entwicklung stattfinden. Idealerweise können die gleichen Werkzeuge und Programmiersprachen eingesetzt werden. Der Vorteil liegt auf der Hand. Das Entwicklungs-Knowhow kann direkt weiterverwendet werden.

Der Einsatz des RAD Servers in Kombination mit der integrierten Entwicklungsumgebung RAD Studio bietet diese Möglichkeit. Front- und Backendentwicklung basieren auf etablierten RAD Technologien und der Verwendung von hochintegrierten Komponenten. Auf diese Weise sind erhebliche Verkürzungen bei der Time-to-Market möglich.

Informationen zu den Produkten

Embarcadero stellt Tools bereit, die die Produktivitätsprobleme von Anwendungsentwicklern lösen. Mit den Produkten kann man Anwendungen aus einer einzigen Codebasis für alle Plattformen entwerfen, erstellen und ausführen.
Mehr als drei Millionen C++- und Delphi-Anwender weltweit setzen auf die preisgekrönten Produkte von Embarcadero, um unternehmenskritische Anwendungen zu liefern.

Um sich grundlegend mit den vielfältigen technologischen Möglichkeiten von Delphi und C++Builder vertraut zu machen, steht eine Testversion zur Verfügung. Diese kann man von www.embarcadero.com/de/products/rad-studio/start-for-free herunterladen und sich auf der Webseite umfassend über die Features informieren.

Testen Sie 30 Tage gratis!