SE - API

Florian Wiese

November 3, 2017

Was ist das überhaupt?

- Application Programming Interface(API) wird meistens als Programmierschnittstelle bezeichnet
- sie sorgt dafür, dass Programme auf Daten und auf Hardware zugreifen können, die hinter der Schnittstelle stehen
- die API dient also zur Weiterverarbeitung von Daten und gibt dem Programm dann entsprechende Daten zurück
- sie ist das Maschinenequivalent zum Userinterface. Sie ist statt von Menschen, von Maschinen lesbar

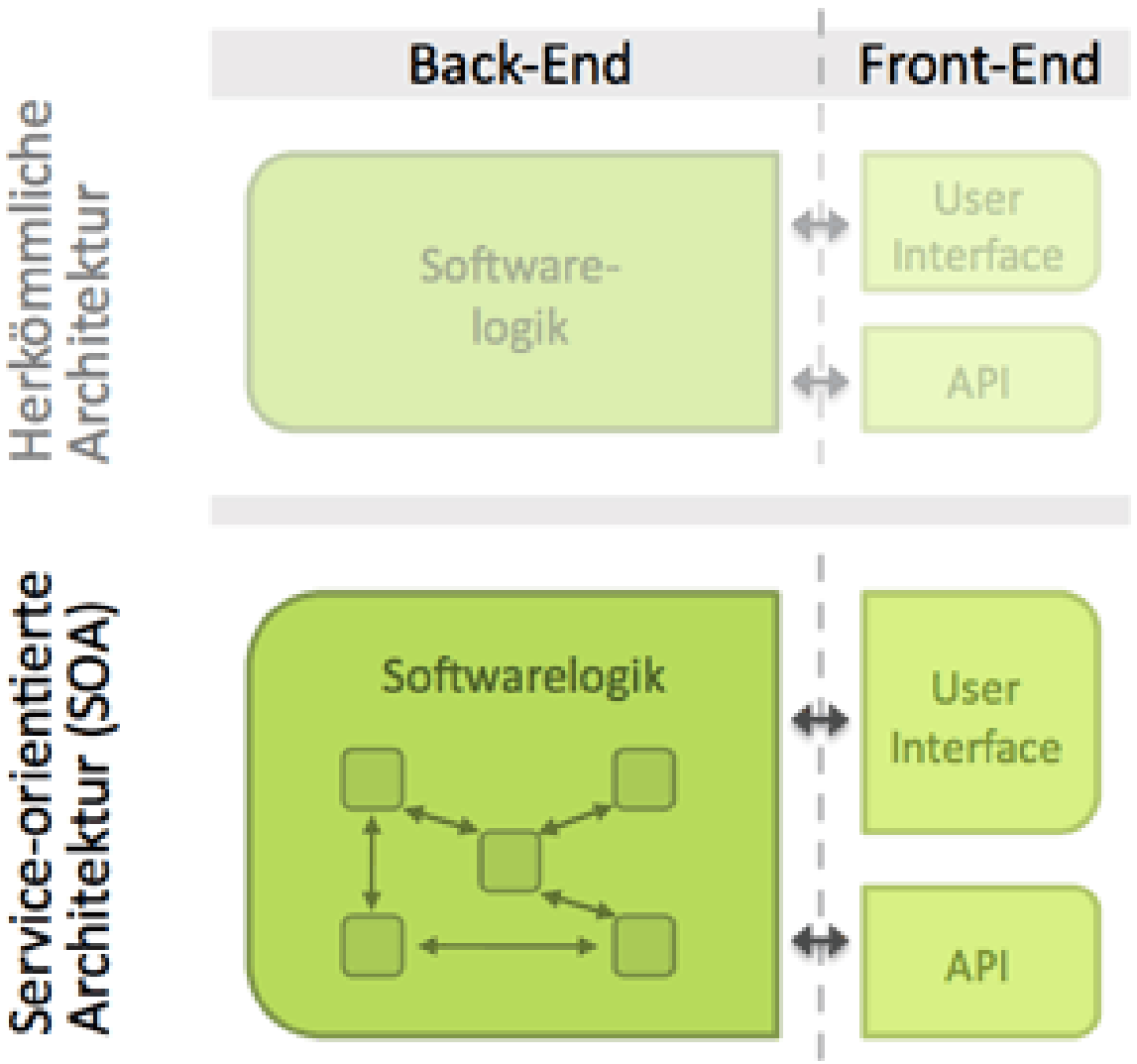
Grundlegendes

- Damit man eine API benutzen kann, sollte idealer Weise eine gute Dokumentation vorliegen
- Um das zu gewährleisten gibt es Standards an die man sich haltem muss
- Diese Standards sind: SOAP, XML-RPC oder REST
- Als Datenformat gibt es JSON und XML

Unterschiedliche Arten von API's

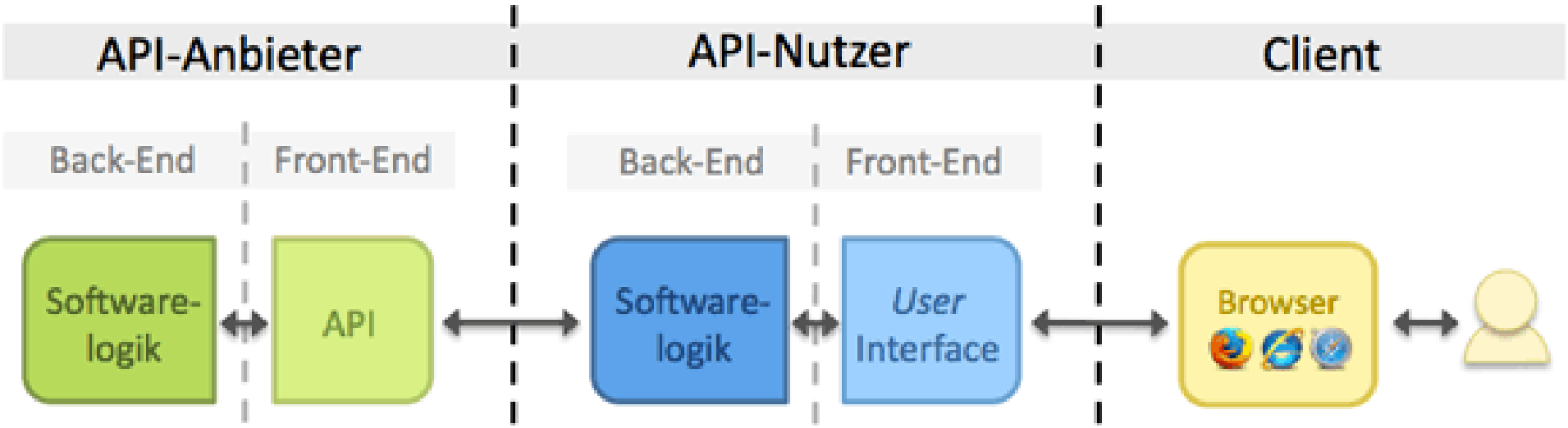
1. Interne API's
2. externe API's
3. Plattform API's
4. Authentifizierungs- und Autorisierungs-APIs

Interne API


Herkömmliche Architektur
Back-End
Front-End
Software-logik
User Interface
API
Service-orientierte Architektur (SOA)
Softwarelogik
User Interface
API

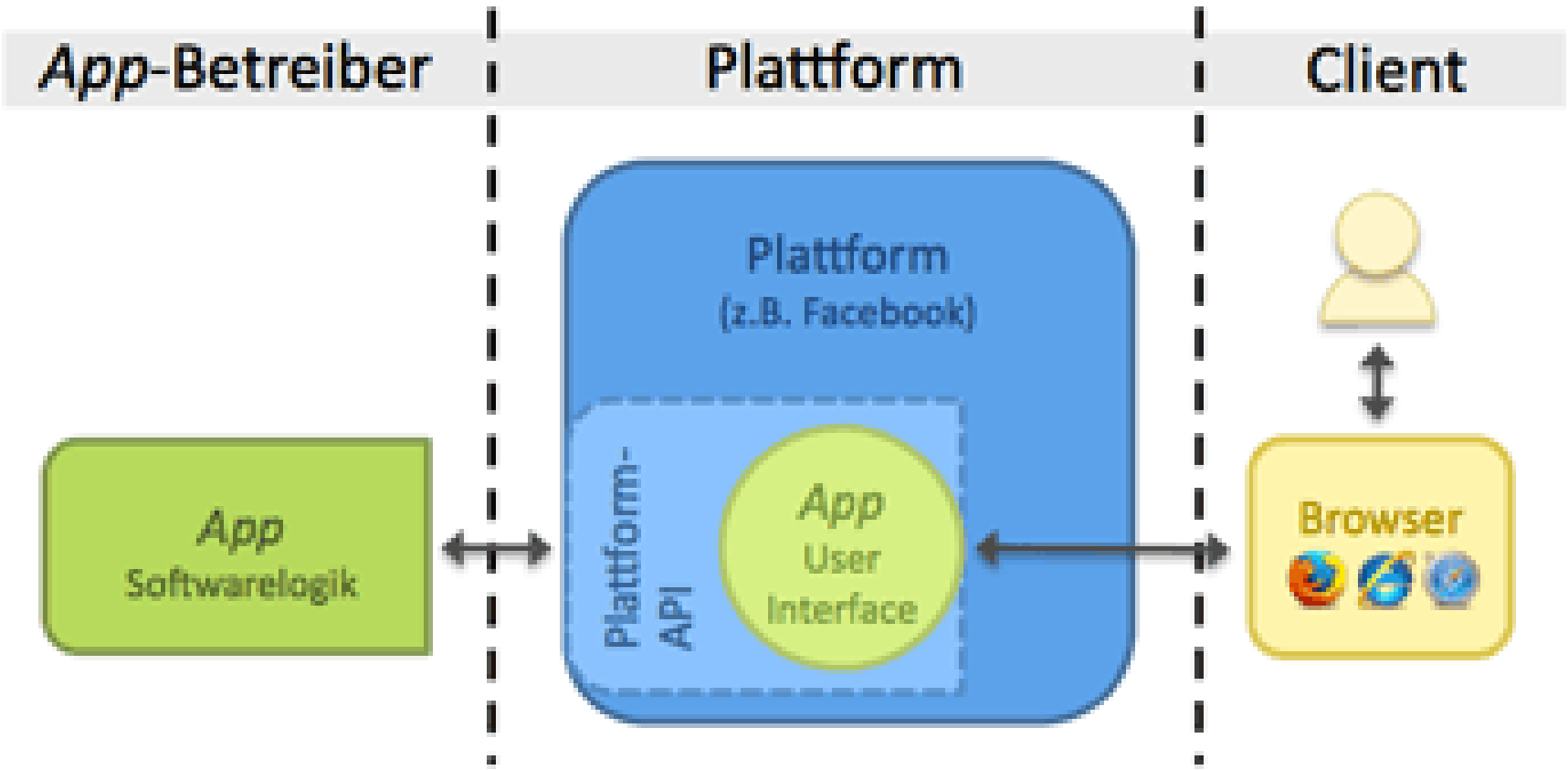
- Je klarer die Abgrenzung der API zum Code, desto besser
- Eine interne API gehört zum guten Ton
- Das steigert die Modularität und damit auch die Wiederverwendbarkeit
- SOA(Service Oriented Architecture) geht so weit, dass man das Gesamtsystem in möglichst viele kleine, unabhängige Teile zerlegt, die dann untereinander kommunizieren

Externe API


API-Anbieter
API-Nutzer
Client
Back-End
Front-End
Back-End
Front-End
Software-logik
API
Software-logik
User Interface
Browser

- Der Client greift mit einem Browser auf ein Interface zu hinter dem die Softwarelogik steht
- Diese Softwarelogik greift dann auf die API zu, die die Daten zurückliefert und dann über das Interface an den User zurückgibt
- Ein typisches Beispiel dafür ist z.B. YouTube

Plattform API's



- Sie sind eine Schnittstelle um die Plattform auch auf anderen Websites zu integrieren und zu betreiben.
- Der User greift über den Browser oder die App auf die API zu, in der auch das Interface steckt
- Damit muss sich ein Drittanbieter nicht mehr um das Interface kümmern, sondern hat einen Standard

Authentifizierungs- und Autorisierungs-APIs

- Sie sind eine besondere Form von WEB-API's, die Nutzer verwalten
- Andere Applikationen nutzen dann diese API um keinen eigenen Nutzerkonten anlegen zu müssen
- Man nutzt zum einloggen die Daten, die bereits hinter der API vorhanden ist
- Der Nutzer muss dafür aber zustimmen Daten an Dritte weiterzugeben

Vorteile von API's

- Komplexe Software wird modularisiert und dadurch einfacher
- dadurch wird die Software wartbarer und auch nicht so fehleranfällig
- Drittanbieter können durch die API auch an der Software mitarbeiten, was das Auslagern von Arbeiten möglich macht
- erhöht dadurch Gesamtattraktivität des Produktes

REST(Representational State Transfer)

Da unser System unter dem Standard REST laufen wird, erkläre ich hier nochmal was das ist.
Über REST weiß der Server was zurückgegeben werden muss. Deswegen muss jede REST Nachricht alle nötigen Informationen enthalten, die wichtig sind. Es werden weder Daten auf dem Client, noch auf dem Server gespeichert.
Das bedeutet, dass die Anfragen geschlossen sind. Das nennt man auch zustandslos.

Jede Information, die eine eigene URL hat, wird als Ressource gekennzeichnet. Das sorgt für eine konstante Adressierbarkeit.
Der Server hat für diese Ressource jetzt verschiedene Ausgabeformate.(JSON XML)
Die Veränderung dieser Ressource sollte man dann über diese Datei vornehmen

Ein paar Grundfunktionen von REST

- GET
- POST
- PUT
- DELETE

GET

GET liefert eine angegebene Ressource zurück.

```
GET warenkorb/6661
```

könnte zum Beispiel folgendes XML Dokument zurückliefern:

```
HTTP/1.1 200 OK Content-Type: text/xml
<?xml version="1.0"?>
<warenkorb xmlns:xlink="http://www.w3.org/1999/xlink">
    <kunde xlink:href="http://shop.oio.de/kunde/5873"> 5873</kunde>
    <position nr="1" menge="5">
        <artikel xlink:href="http://shop.oio.de/artikel/4501" nr="4501">
            <beschreibung>Dauerlutscher</beschreibung>
        </artikel>
    </position>
    <position nr="2" menge="2">
        <artikel xlink:href="http://shop.oio.de/artikel/5860" nr="5860">
            <beschreibung>Earl Grey Tea</beschreibung>
        </artikel>
    </position>
</warenkorb>
```

Hier kann man sehen woraus der Warenkorb besteht. Der Client könnte diese XML Datei jetzt verarbeiten und sehen was im Warenkorb enthalten ist.

POST

Der POST Befehl legt unter einer Ressource noch eine Unterressource an. Dies würde, wenn man beim Beispiel des Warenkorbs bleibt, diesem zum Beispiel einen weiteren Artikel hinzu.

```
POST /warenkorb/5873
artikelnummer=961
```

Würde dem Warenkorb jetzt den Artikel 961 hinzufügen

PUT

Mit PUT legt man neue Ressourcen auf dem Server an. Man legt sozusagen die XML,JSON Datei auf den Server hoch. Dieser legt dann eine neue Ressource an, die eine eigene URL bekommt.

DELETE

DELETE löscht die, wie der Name schon sagt, den Ressource den man angibt.

```
DELETE /artikel/6005
```

löscht einen Artikel. Beispielsweise in einem Warenkorb

Quellen

- https://www.dev-insider.de/was-ist-eine-api-a-583923/
- https://www.gruenderszene.de/allgemein/web-apis-ein-nicht-technischer-erklarungsversuch
- https://www.oio.de/public/xml/rest-webservices.htm
- https://de.wikipedia.org/wiki/Representational_State_Transfer