

Datenbank Verwaltungstool

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für Medientechnik

Eingereicht von:

David Altenhofer

Sami Abbas Ali

David Precup

Betreuer:

Michael Palitsch-Infanger

Projektpartner:

Sami Abbas Ali, David Precup, Doka GmbH

Leonding, Juli 2022

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Leonding, April 2022

S. Schwammal & S. Schwammal

Abstract

The Doka company offers software packages to their customers. Here is the administration still very much in need of improvement. The process of adding new software packages, works as follows: creating new packages, editing and deleting takes place directly via the database, which means that there is no administration program to date. Since everything runs through the database, it becomes clean work prevented. That's why we were commissioned by the Doka company to create a Database Management System based on the JavaScript library React. Before some terms need to be explained:

It should be possible to manage the following tables that have these properties:

- InstallablePackage is the main table, consists of several Installables and can have a description (InstallablePackagesDescriptions) in multiple languages.
- An Installable has an InstallableSyncTemplate and can have an InstallableExecutablePaths.

The program should have the following functions:

- Viewing, installing, adding, editing and deleting so-called Installables / InstallablePackages / InstallableSyncTemplates / InstallableExecutablePaths
- Exporting the current configuration
- Importing a JSON file, where to see the difference of the imported file and the current configuration
- Reset the cache in the backend



Abbildung 1: ERD

Zusammenfassung

Die Firma Doka bietet Softwarepakete an deren Kunden an. Dabei ist das Verwalten noch sehr verbesserungswürdig. Der Prozess des hinzufügens von neuen Softwarepaketen, läuft wie folgt ab: Das Erstellen von neuen Paketen, das Editieren sowie das Löschen erfolgt direkt über die Datenbank, das bedeutet, dass es kein Verwaltungsprogramm bis dato gibt. Dadurch, dass alles über die Datenbank abläuft, wird das saubere Arbeiten verhindert. Deshalb wurden wir von der Firma Doka dazu beauftragt, ein Verwaltungssystem basierend auf der JavaScript library React zu programmieren. Davor müssen einige Begriffe noch erklärt werden:

Folgende Tabellen sollen verwaltet werden können, die diese Eigenschaften haben:

- InstallablePackage ist die wichtigste Tabelle, besteht aus mehreren Installables und kann Beschreibung (InstallablePackagesDescriptions) auf mehreren Sprachen haben.
- Ein Installable hat ein InstallableSyncTemplate und kann ein InstallableExecutablePaths.

Dabei soll das Programm folgende Funktionen haben:

- Das Anzeigen, Installieren, Hinzufügen, Bearbeiten und Löschen von sogenannten Installables / InstallablePackages / InstallableSyncTemplates / InstallableExecutablePaths
- Das Exportieren der aktuellen Konfiguration
- Importieren eines JSON Dateis, wo die Differenz der importierten Datei und der aktuellen Konfiguration zu sehen ist
- Reseten des Cache im Backend

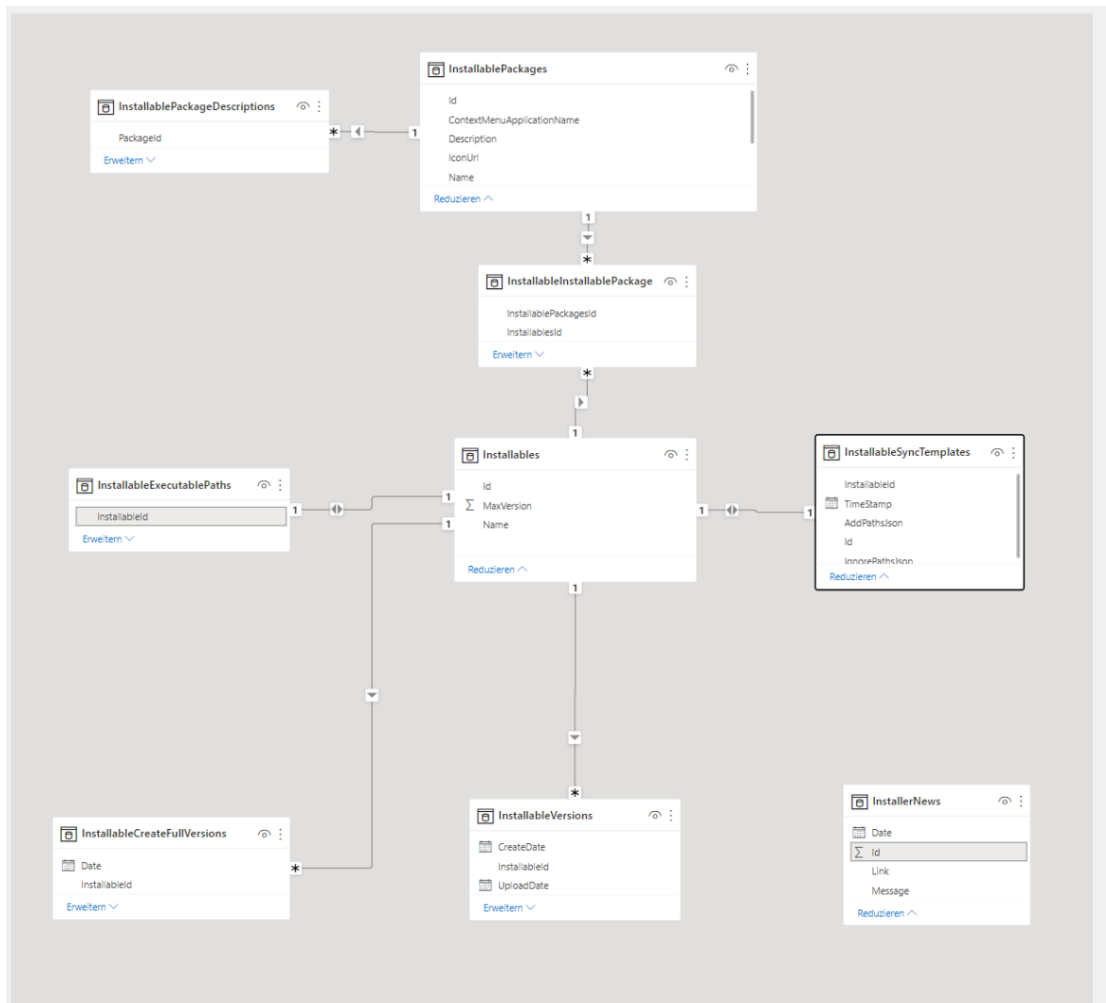


Abbildung 2: ERD

Inhaltsverzeichnis

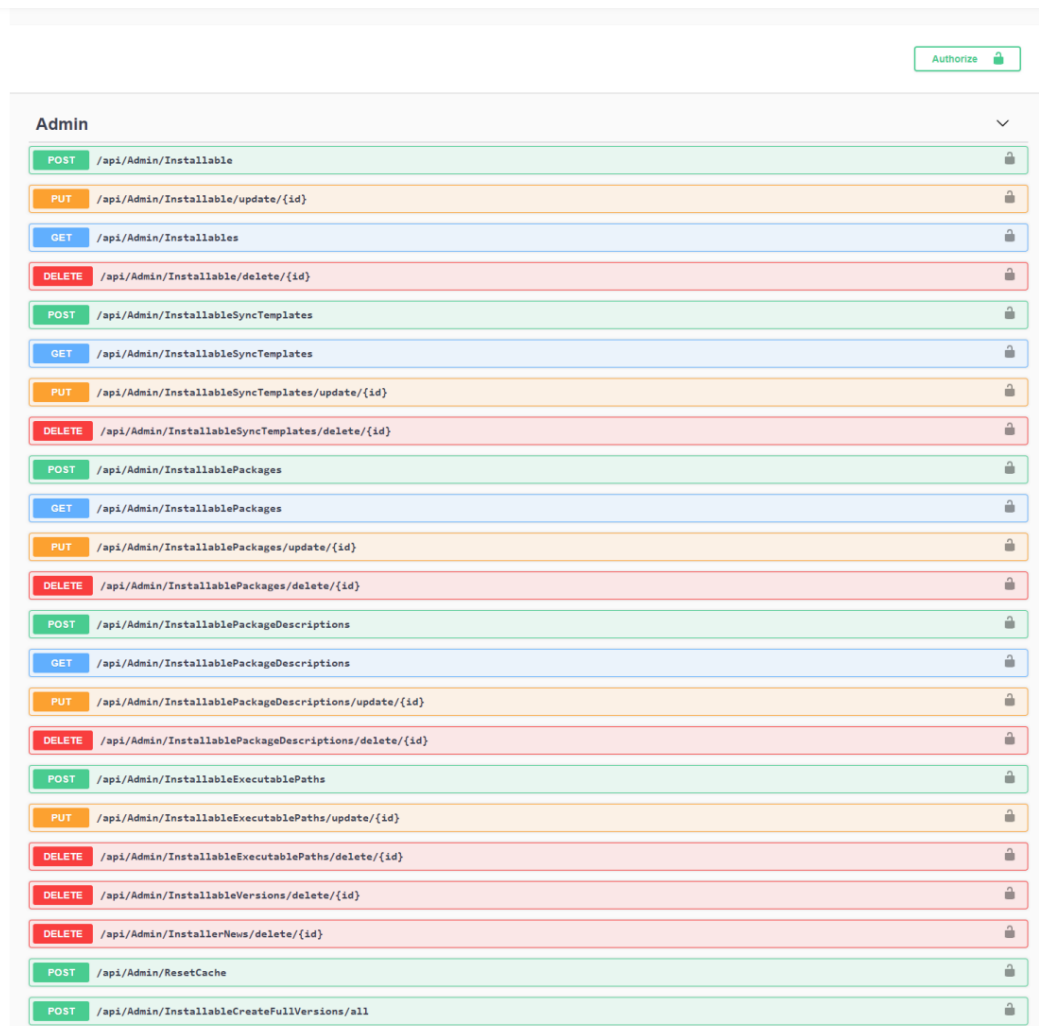
1	Einleitung	1
1.1	Ausgangssituation	1
1.2	Aufgabenstellung	4
1.3	Zielsetzung	4
2	Umfeldanalyse	5
3	Technologien	6
3.1	React	6
3.2	Tailwind CSS	8
3.3	Material UI	9
3.4	Swagger UI	9
3.5	HTTP-Request / REST	9
3.6	OAuth	9
3.7	OIDC	9
3.8	Dev Ops	9
4	Umsetzung	10
5	Zusammenfassung	12
	Literaturverzeichnis	VII
	Abbildungsverzeichnis	VIII
	Tabellenverzeichnis	IX
	Quellcodeverzeichnis	X
	Anhang	XI

1 Einleitung

1.1 Ausgangssituation

Seit Jahren müssen die Mitarbeiter der Firma die Datenbank Einträge sehr umständlich ändern, da kein User Interface dafür vorhanden ist. Unser Ziel war es, genau das zu Ändern. Ich habe gemeinsam mit meinem Team ein Web-Frontend erstellt, welches das Hinzufügen, Bearbeiten und Löschen der Einträge in der Datenbank um ein vielfaches vereinfacht. Nun kann auf das unpraktische Verwenden von Swagger verzichtet werden und auf ein paar wenige Klicks reduziert werden.

Gegeben war bereits ein fertiges Backend mit einer API



Admin		▼
POST	/api/Admin/Installable	🔒
PUT	/api/Admin/Installable/update/{id}	🔒
GET	/api/Admin/Installables	🔒
DELETE	/api/Admin/Installable/delete/{id}	🔒
POST	/api/Admin/InstallableSyncTemplates	🔒
GET	/api/Admin/InstallableSyncTemplates	🔒
PUT	/api/Admin/InstallableSyncTemplates/update/{id}	🔒
DELETE	/api/Admin/InstallableSyncTemplates/delete/{id}	🔒
POST	/api/Admin/InstallablePackages	🔒
GET	/api/Admin/InstallablePackages	🔒
PUT	/api/Admin/InstallablePackages/update/{id}	🔒
DELETE	/api/Admin/InstallablePackages/delete/{id}	🔒
POST	/api/Admin/InstallablePackageDescriptions	🔒
GET	/api/Admin/InstallablePackageDescriptions	🔒
PUT	/api/Admin/InstallablePackageDescriptions/update/{id}	🔒
DELETE	/api/Admin/InstallablePackageDescriptions/delete/{id}	🔒
POST	/api/Admin/InstallableExecutablePaths	🔒
PUT	/api/Admin/InstallableExecutablePaths/update/{id}	🔒
DELETE	/api/Admin/InstallableExecutablePaths/delete/{id}	🔒
DELETE	/api/Admin/InstallableVersions/delete/{id}	🔒
DELETE	/api/Admin/InstallerNews/delete/{id}	🔒
POST	/api/Admin/ResetCache	🔒
POST	/api/Admin/InstallableCreateFullVersions/all	🔒

Abbildung 3: Swagger UI - Endpoints

POST	/api/Admin/InstallableCreateFullVersions/{installableId}	🔒
GET	/api/Admin/InstallableCreateFullVersions	🔒
InstallableQuery		
GET	/api/InstallableQuery/InstallableVersions/GetVersionInfos/{culture}	🔒
GET	/api/InstallableQuery/InstallableVersions/{id}	🔒
GET	/api/InstallableQuery/InstallableExecutablePaths	🔒
GET	/api/InstallableQuery/InstallerNews	🔒
InstallableUpload		
POST	/api/InstallableUpload/RequestUpload	🔒
POST	/api/InstallableUpload/NotifyUpload	🔒
GET	/api/InstallableUpload/InstallableSyncTemplates	🔒
GET	/api/InstallableUpload/GetInstallableVersionUploadSettingQuery/{installableId}	🔒
POST	/api/InstallableUpload/InstallerNews	🔒
Schemas		
InstallableDto >		
InstallablePackageDescriptionDto >		
InstallablePackageDto >		
InstallableSyncTemplateDto >		
InstallableVersionDto >		
InstallableVersionInfo >		
InstallableVersionPackageDto >		
InstallableVersionPackageInfo >		
InstallerNewsDto >		
SyncPathDto >		

Abbildung 4: Swagger UI - Endpoints

1.2 Aufgabenstellung

Um den Umgang mit der Datenbank für alle Mitarbeiter zu vereinfachen, soll ein leicht verständliches, intuitives Interface entwickelt werden. Die Web-Anwendung darf nur für autorisierte Benutzer in der Abteilung der Firma verfügbar sein. Deswegen ist eine Authentifizierungslogik zu implementieren. Erst nach erfolgreichen Anmelden mit einem gültigen Doka-Account, soll die Website erreichbar sein und Änderungen in der Datenbank durchgeführt werden können.

1.3 Zielsetzung

Die Web-Anwendung soll in einem ansprechenden Design entwickelt werden und ein leicht verständlicher unkomplizierter Umgang muss auch für Nicht-Fachleute gegeben sein.

2 Umfeldanalyse

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula. Citing [1] properly.

Was ist eine Globally Unique Identifier (GUID)? Eine GUID kollidiert nicht gerne.

Kabellose Technologien sind in abgelegenen Gebieten wichtig [2].

3 Technologien

3.1 React

React ist eine **JavaScript-Programmbibliothek** zur Erstellung von webbasierten Benutzeroberflächen. Jede React-Webanwendung besteht aus wiederverwendbaren Komponenten, die Teile der Benutzeroberfläche bilden – wir können eine separate Komponente für unsere Navigationsleiste haben, eine für die Fußzeile, eine andere für den Hauptinhalt und so weiter.

Diese wiederverwendbaren Komponenten machen die Entwicklung einfacher, weil man wiederkehrenden Code nicht wiederholen muss. Man muss nur die Logik erstellen und die Komponente in jeden Teil des Codes importieren, in dem sie benötigt wird. React ist außerdem eine Single-Page-Anwendung. Anstatt also jedes Mal, wenn eine neue Seite gerendert werden soll, eine Anfrage an den Server zu senden, wird der Inhalt der Seite direkt von den React-Komponenten geladen. Das führt zu einem schnelleren Rendering ohne Neuladen der Seite.

In den meisten Fällen wird für die Erstellung von React-Apps die Syntax JSX (JavaScript XML) verwendet, eine Syntaxerweiterung von JavaScript. Damit kann man die Logik von JavaScript und die Logik der Benutzeroberfläche auf einzigartige Weise kombinieren. Mit JSX muss kein DOM integriert werden, weil man einfach Methoden wie **document.getElementById**, **querySelector** und andere DOM-Manipulationsmethoden verwendet. Die Verwendung von JSX ist zwar nicht obligatorisch, aber sie macht die Entwicklung von React-Anwendungen einfacher.

Im oberen Code-Beispiel wird eine funktionale React-Komponente verwendet, um ein Stück Text im Browser darzustellen. Der Name der Komponente ist **App**. Es wird eine Variable vor der Funktion **render()** erstellt. Diese Variable wird mit geschweiften Klammern an das Markup übergeben. Das ist kein HTML, sondern die Syntax für das Schreiben von Code mit JSX.

```
function App() {  
  const greetings = "Hello World";  
  return (  
    <div className="App">  
      <h1> {greetings} </h1>  
    </div>  
  );  
}
```

Abbildung 5: JSX-Example in React [2]

3.2 Tailwind CSS

Tailwind CSS ist ein Utility-First-CSS-Framework. Es stellt vorgefertigte CSS-Klassen bereit, die verwendet werden können, um HTML-Elemente schnell und einfach zu gestalten. Im Vergleich mit anderen CSS-Frameworks, die vorgefertigte UI-Komponenten bereitstellen, fokussiert sich Tailwind mehr auf Low-Level-Bausteine wie Abstände (Paddings und Margins), Schriftarten und Farbe, um benutzerdefinierte Styles zu erstellen.

Entwickler können vorgefertigte CSS-Klassen hinzufügen, anstatt für jedes Element benutzerdefiniertes CSS zu schreiben. Das beschleunigt nicht nur den Entwicklungsprozess, sondern kann sich auch positiv auf die Performance auswirken. Die Dokumentation von Tailwind CSS ist umfangreich und die große Community erleichtert den Einstieg und bietet Unterstützung.

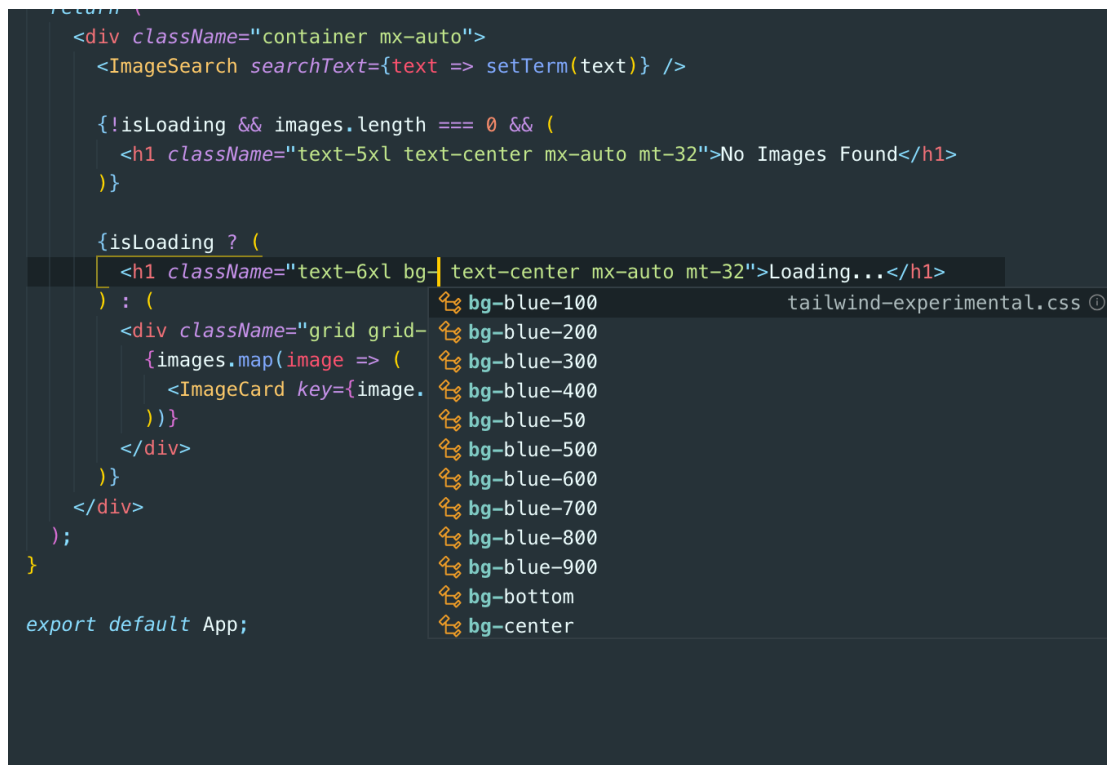


Abbildung 6: Tailwind - vorgefertigte CSS-Klassen [2]

3.3 Material UI

3.4 Swagger UI

3.5 HTTP-Request / REST

3.6 OAuth

3.7 OIDC

3.8 Dev Ops

3.8.1 Deeper

Nicht mehr im Inhaltsverzeichnis.

4 Umsetzung

Siehe tolle Daten in Tab. 1.

Siehe und staune in Abb. 7. Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetur at, consectetur sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

	Regular Customers	Random Customers
Age	20-40	>60
Education	university	high school

Tabelle 1: Ein paar tabellarische Daten

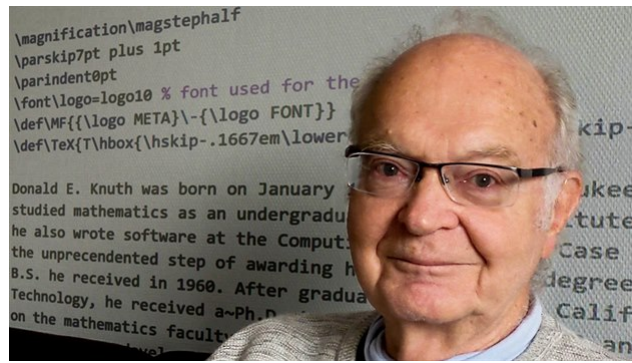


Abbildung 7: Don Knuth – CS Allfather

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetur a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetur. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus. Dann betrachte den Code in Listing 1.

Listing 1: Some code

```

1  # Program to find the sum of all numbers stored in a list (the not-Pythonic-way)
2
3  # List of numbers
4  numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]
5
6  # variable to store the sum
7  sum = 0
8
9  # iterate over the list
10 for val in numbers:
11     sum = sum+val
12
13 print("The sum is", sum)

```

5 Zusammenfassung

Aufzählungen:

- Itemize Level 1
 - Itemize Level 2
 - Itemize Level 3 (vermeiden)
- 1. Enumerate Level 1
 - a. Enumerate Level 2
 - i. Enumerate Level 3 (vermeiden)

Desc Level 1

Desc Level 2 (vermeiden)

Desc Level 3 (vermeiden)

Literaturverzeichnis

- [1] P. Rechenberg, G. Pomberger *et al.*, *Informatik Handbuch*, 4. Aufl. München – Wien: Hanser Verlag, 2006.
- [2] Association for Progressive Communications, „Wireless technology is irreplaceable for providing access in remote and scarcely populated regions,” 2006, letzter Zugriff am 23.05.2021. Online verfügbar: <http://www.apc.org/en/news/strategic/world/wireless-technology-irreplaceable-providing-access>

Abbildungsverzeichnis

1	ERD	II
2	ERD	IV
3	Swagger UI - Endpoints	2
4	Swagger UI - Endpoints	3
5	JSX-Example in React [2]	7
6	Tailwind - vorgefertigte CSS-Klassen [2]	8
7	Don Knuth – CS Allfather	11

Tabellenverzeichnis

1	Ein paar tabellarische Daten	10
---	--	----

Quellcodeverzeichnis

1	Some code	11
---	---------------------	----

Anhang