

bfw

Unternehmen für Bildung.

Benutzerschnittstellen gestalten und entwickeln

Modul: ITBS FIA K
Version: 1.0

Dezember 2022

Erstellt von: IT-Ausbildungsabteilung

Modul: ITBS FIA K Benutzerschnittstellen gestalten

Version, Datum: 1. Dezember 2022

Erstellt von: IT_Ausbildungsabteilung

Alle Rechte vorbehalten. Kein Teil des Werkes darf in irgendeiner Form (Druck, Fotokopie oder einem anderen Verfahren) ohne Genehmigung des Herausgebers reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.
Die Modulunterlage und alle ihre Bestandteile wurden mit großer Sorgfalt erstellt und geprüft. Dennoch können Fehler nicht vollkommen ausgeschlossen werden. Der Herausgeber und die Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

Literaturhinweis

Titel: IT Berufe Fachstufe II FIAE & FIDPA

ISBN: 987-3-14-220072-9

Verlag: Westermann Gruppe

Inhaltsverzeichnis

Inhaltsverzeichnis 3
1. Anforderungen an eine Benutzerschnittstelle ermitteln 7
1.1. Den Prozess der Anforderungsspezifikation präsentieren 8
1.2. Arbeitsabläufe und Geschäftsprozesse im Rahmen einer Ist-Analyse beschreiben 12
1.3. Anforderungskatalog erstellen 15
1.3.1. Funktionale und nicht funktionale Anforderungen an Benutzerschnittstellen 16
1.3.2. Anforderungskatalog für das Beispielszenario 17
2. Ein Softwareprodukt designen 19
2.1. Benutzerschnittstellen in den Kontext der Softwarearchitektur einordnen 20
2.2. Architekturmuster unterscheiden 23
2.2.1. Drei-Schichten-Architektur 24
2.2.2. Client-Server-Architektur 25
2.2.3. Model View Controller (MVC) 25
3. Benutzerschnittstellen funktionsgerecht und ergonomisch konzipieren 28
3.1. Benutzerschnittstellen unterscheiden und Zukunftstrends präsentieren 28
3.2. Kriterien zur Gestaltung und Bewertung von Benutzerschnittstellen präsentieren 32
3.3. Den Designprozess und Designwerkzeuge präsentieren 36
3.4. Grafische Benutzerelemente einer GUI unterscheiden und allgemeine Gestaltungskriterien präsentieren 39
3.4.1. Grundlegende GUI-Elemente 39
3.4.2. Allgemeine Gestaltungsregeln 41
3.4.3. Weitere Gestaltungsaspekte 42
3.4.4. Bedienung von grafischen Oberflächen mit anderen Interaktionsformen 45
3.4.5. Text-to-Speech Systeme 45
4. Benutzerschnittstellen in einer Programmiersprache implementieren 47
4.1. Bibliotheken und Frameworks auswählen 48
4.2. Benutzerschnittstellen in Python realisieren 51
4.2.1. GUI-Programmierung 51
4.2.2. Umsetzung der Beispielaufgabe 57
4.2.3. Sprachausgabe 61
4.3. Benutzerschnittstellen in Java realisieren 64
4.3.1. GUI-Programmierung 64

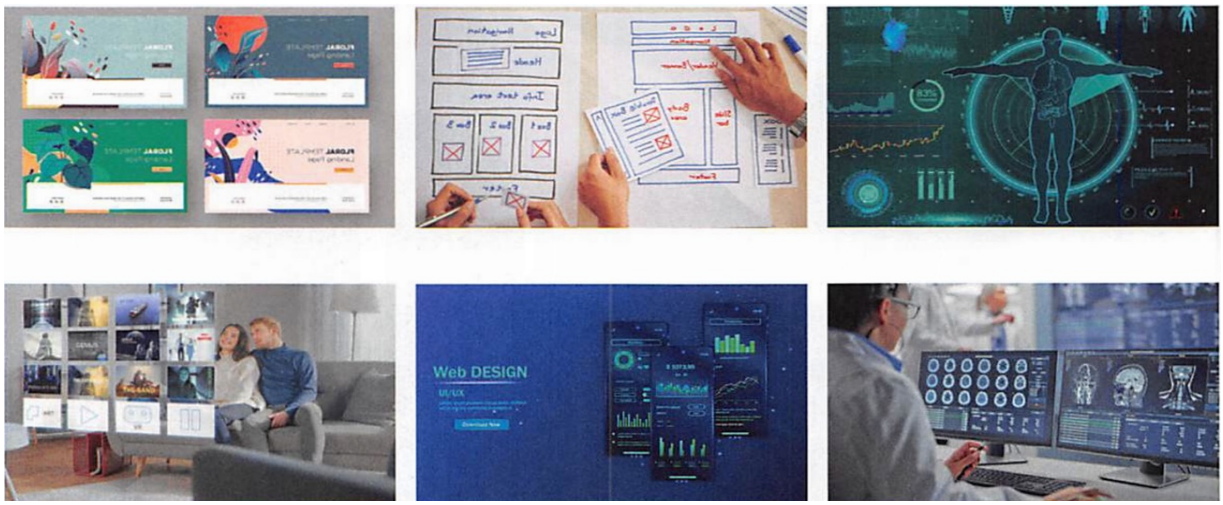
4.3.2. Umsetzung der Beispielaufgabe 73
4.3.3. Sprachausgabe 77
4.4. Webbasierte Benutzerschnittstellen realisieren 81
4.4.1. HTML 81
4.4.2. CSS 86
4.4.3. Navigation und Struktur einer Webseite 89
4.4.4. Formulare und JavaScript 90
5. Benutzerschnittstellen testen 97
5.1. Grafische Benutzerschnittstelle testen 98
5.2. Das Testen der User Experience beschreiben 103
6. Benutzerschnittstellen dokumentieren 106

Die Digitalisierung erfasst mehr und mehr unseren Alltag. Für Nutzer ist es wichtig, dass sie einen unkomplizierten Zugang zu digitalen Angeboten erhalten. Das wird über entsprechende Benutzerschnittstellen erreicht. Dieser Bereich



umfasst auch die Zugänglichkeit und Bedienbarkeit für ältere Menschen oder Menschen mit Behinderung. Deswegen ist eine ergonomische und funktionsgerechte Benutzerschnittstelle, welche auf die entsprechende Zielgruppe ausgerichtet ist, eine wichtige Grundvoraussetzung für die Akzeptanz eines Softwareproduktes.

Um solche Benutzerschnittstellen umzusetzen, bedarf es mehrerer Schritte. Zunächst muss der Prozess, welcher digital abgebildet werden soll, genau analysiert werden. Aus dieser Analyse werden die Anforderungen an das zukünftige Softwareprodukt abgeleitet. Auf Grundlage dieser Anforderungen werden danach u.a. die Benutzerschnittstellen dann geplant und implementiert.



Bei der Gestaltung der Benutzerschnittstellen spielen viele Faktoren eine Rolle. Diese reichen von der Art der Benutzerschnittstelle, über das UX Design bis hin zu Fragen der barrierefreien Gestaltung. In vielen Fällen müssen auch

mehrere, sich stark unterscheidende Benutzerschnittstellen für verschiedene Zielgruppen einer Software entwickelt werden. Für die Umsetzung der Benutzerschnittstellen stehen in der Regel verschiedene Frameworks zur Verfügung, deren Vor- und Nachteile sorgfältig gegeneinander abgewogen werden müssen.

Die Inhalte dieses Moduls werden an verschiedenen Beispielen verdeutlicht. Die Entwicklung, das Testen und die Dokumentation einer Benutzerschnittstelle für die Liegeplatzverwaltung eines „Yachthafen Resorts" dient dabei als Grundlage:



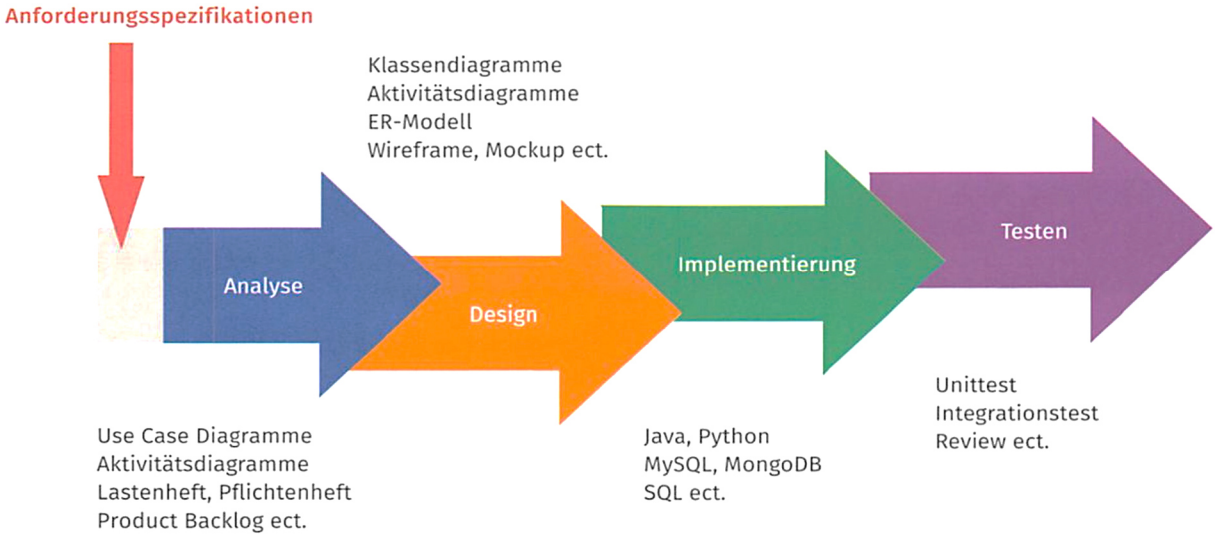
1. Anforderungen an eine Benutzerschnittstelle ermitteln

➢ **Sie sollen Ihr Basiswissen zur Ermittlung der Anforderungen an ein Softwareprodukt festigen und erweitern.**



Ein erster Schritt bei der Umsetzung von Softwareprojekten ist die Erfassung der Kundenbedürfnisse. Diese zu kennen und zu erfüllen ist die Grundlage für den Erfolg eines Unternehmens. Auf den ersten Blick scheint die Erfassung der Kundenwünsche relativ einfach zu sein: Man muss den Kunden doch einfach nur fragen, was er haben möchte. Leider erweist sich in der Praxis, dass die Bestimmung der Kundenerwartungen und der sich daraus ergebenden Anforderungen an die Software, weitaus schwieriger ist als zunächst angenommen.

Die Erfahrung zeigt, dass es den meisten Kunden eher schwerfällt, die Anforderungen an die zukünftige Software genau zu formulieren. Sie wissen meist nur ungefähr, was die Software leisten soll. Das Spektrum ihrer Erwartungen und Anforderungen kann von leicht zu erfassenden Punkten bis hin zu Kriterien verlaufen, welche der Kunde zwar hat und voraussetzt, aber nicht explizit erwähnt. So kann ein Projekt durch unzureichende, falsche oder erst gar nicht definierte Anforderungen scheitern. Deswegen wird in der Analysephase des Softwareentwicklungsprozesses eine genaue Anforderungsspezifikation vorgenommen (siehe auch Jahrgangsband 1, Modul 5).



Je nach Größe des Projektes kann die Bestimmung der Anforderungen an ein Softwareprodukt ein einmaliger oder ein sich wiederholender Vorgang sein. Bei größeren Neuentwicklungen werden zunächst die Anforderungen an das Gesamtprojekt ermittelt und danach Schritt für Schritt für die einzelnen Bereiche verfeinert, z.B. Benutzerschnittstellen oder bestimmte Module. In anderen Fällen müssen bestehende Anwendungen bloß um neue Module erweitert werden, z. B. um eine neue Benutzerschnittstelle. Auch in diesem Fall wird zunächst eine Anforderungsspezifikation vorgenommen, auch wenn diese bei Weitem nicht so umfangreich ausfallen wird wie bei einer Neuentwicklung.

Bei einer Anforderungsspezifikation werden zunächst im Rahmen einer Ist-Analyse die bestehenden Prozesse und deren Informationsfluss analysiert. Auf Grundlage dieser Analyse werden danach die Anforderungen an die zu entwickelnde Software bestimmt und festgehalten, z. B. in Lastenheften oder Product Backlogs. Diese bilden dann wiederum die Grundlage für das Design der Software, z.B. die Softwarearchitektur oder die Benutzeroberfläche.

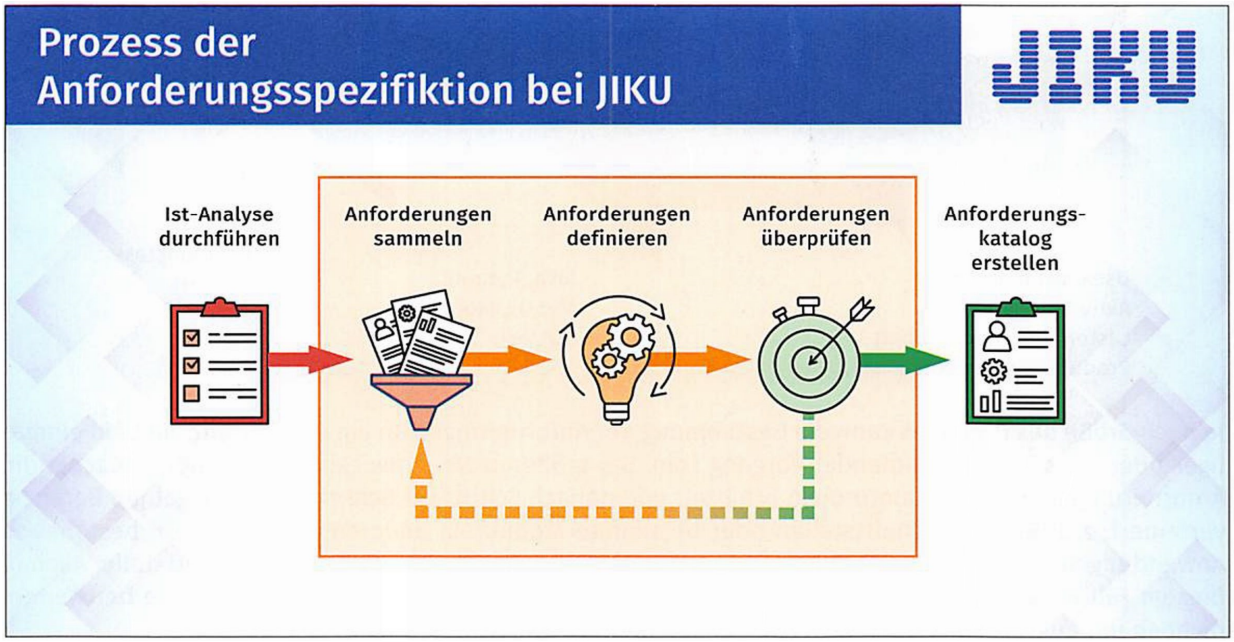
1.1. Den Prozess der Anforderungsspezifikation präsentieren

➢ **Sie sollen sich die einzelnen Prozessschritte der Anforderungsspezifikation erarbeiten und Ihr Ergebnis präsentieren.**



Am Anfang eines jeden Softwareentwicklungsprozesses steht eine Anforderungsspezifikation. Dabei spielt es zunächst keine Rolle, nach welchem Vorgehensmodell die Software entwickelt werden soll. Allerdings hängt es später stark vom Vorgehensmodell ab, wann und wie häufig eine Anforderungsspezifikation vorgenommen wird. Beim klassischen Wasserfallmodell (siehe auch Jahrgangsband 1, Modul 5) wird beispielsweise eine Anforderungsanalyse nur einmal und am Anfang des Projektes durchgeführt. Bei agilen Vorgehensmodellen hingegen, z. B. Scrum (siehe auch Jahrgangsband 2, Modul 8), wird diese jeweils vor dem nächsten Entwicklungszyklus wiederholt und die vorliegenden Anforderungen werden ergänzt, angepasst und verfeinert.

Anforderungsspezifikation
Der Prozess der Anforderungsspezifikation umfasst das Ermitteln, Analysieren, Spezifizieren und Validieren (Überprüfen) aller Anforderungen an das zu entwickelnde Softwaresystem.



Die Anforderungsspezifikation selbst ist ebenfalls ein Prozess, welcher in verschiedene Schritte unterteilt werden kann, deren Ergebnis am Ende ein Anforderungskatalog ist. Die Form und Bezeichnung dieses Katalogs hängt wieder stark von dem verwendeten Vorgehensmodell ab. Dieser kann z. B. ein Lastenheft beim Wasserfallmodell oder ein Product Backlog bei Scrum sein, wobei ein Product Backlog nie vollständig ist und während der Entwicklung der Software regelmäßig durch neue Anforderungsspezifikationen angepasst wird.

Prozessschritte der Anforderungsspezifikation

Ist-Analyse
Die Ist-Analyse ist ein erster wichtiger Schritt im Prozess der Anforderungsspezifikation. Zunächst geht es um die Schaffung eines Gesamtüberblicks über alle für das Projekt wichtigen Abläufe und Prozesse, welche im Anschluss genau definiert werden. Ziel der Ist-Analyse ist es, vorhandene Defizite und Optimierungspotenziale bei den bestehenden Prozessen aufzudecken. Durch eine genaue Ist- Analyse soll sichergestellt werden, dass nur wirklich benötigte Funktionalitäten des zukünftigen Softwaresystems im Anforderungskatalog erfasst werden.

Folgende Punkte können in einer Ist-Analyse untersucht werden:

- Aufgabe des Prozesses: Art und Umfang aller anfallenden Aufgaben und deren Besonderheiten
- Struktur des Prozesses: die Organisation des Prozessablaufes
- Kommunikation innerhalb des Prozesses: Ablauf und Vorrichtungen, welche zur Kommunikation genutzt werden
- Prozessdaten: Art und Umfang der im Prozess zu verarbeitenden Daten
- Prozessdokumentation: Art und Umfang der Dokumentation des Prozesses
- Schwachstellen des Prozesses: bekannte Mängel, Behinderungen oder Redundanzen des Prozesses

Anforderungen sammeln
In diesem Schritt werden die Anforderungen an das zu entwickelnde Softwaresystem zusammengetragen. Die Grundlage dafür bildet die Ist-Analyse. Dabei sollten auch weitere Anforderungen berücksichtigt werden, welche z. B. vom Management oder zusätzlichen Nutzern kommen. Das Ziel dieses Schrittes sollte eine vollständige Bedarfsübersicht sein, deren Inhalte beispielsweise in folgende Kategorien unterteilt werden können:

- funktionale Anforderungen
- technische Anforderungen
- Organisatorische Anforderungen
- allgemeine Kriterien

Anforderungen definieren
Aus den gesammelten Anforderungen werden nun konkrete Anforderungen an das Softwareprodukt definiert und ausformuliert. Dabei sind u.a. folgende Überlegungen einzubeziehen:

- Können die Anforderungen überhaupt erfüllt werden?
- Welche technischen Einschränkungen gibt es?
- Welche Kosten fallen an und welche Ressourcen sind nötig?
- Welche Risiken existieren?

Die konkreten Anforderungen werden nun Softwarekriterien zugeordnet und in funktionale und nicht funktionale Anforderungen klassifiziert.

Anforderungen überprüfen
In vielen Fällen können nicht alle gestellten Anforderungen berücksichtigt werden. Das mag zum einen daran liegen, dass sie technisch nicht machbar oder in dem gegebenen Zeitraum nicht umsetzbar sind. Auf der anderen Seite sind nicht immer alle Anforderungen unbedingt für die Kernfunktion der Software notwendig. Daher sollten die Anforderungen dahingehend überprüft werden, ob sie im Anforderungskatalog verbleiben und dort dann priorisiert werden. Mögliche Priorisierung wäre:

- • nicht erforderlich • nice-to-have
- unbedingt erforderlich

Anforderungskatalog fertigstellen und genehmigen
In diesem Schritt wird der Anforderungskatalog fertiggestellt. Dieser kann sich, je nach Vorgehensmodell im Aufbau und in der Bezeichnung unterscheiden (z. B. Lastenheft, Product Backlog). In einigen Fällen ist der Inhalt dann feststehend, in anderen wird dieser in sich wiederholenden Durchläufen angepasst bzw. erweitert. Zu einem Anforderungskatalog sollten in jedem Fall alle Beteiligten ihre Zustimmung geben, bevor auf dessen Grundlagen das Projekt weiter vorgesetzt wird.

Für die Erfassung der notwendigen Informationen bei der Anforderungsanalyse gibt es verschiedene Methoden. Einige davon werden nachfolgend genauer betrachtet.

Methoden zur Erhebung des Istzustandes und der Anforderungen	
Interview	Beim Interview werden die Beteiligten in Form eines mündlichen Fragebogenszu einem oder mehreren Themenkomplex befragt. **Vorteile:** • persönlicher Kontakt fördert die Bereitschaft zur Auskunft • gestellte Fragen werden mit großer Wahrscheinlichkeit auch beantwortet • Unklarheiten können direkt geklärt werden **Nachteile:** • hoher zeitlicher Aufwand • subjektive Prägung der Antworten • erschwerte Erfassung von Faktendaten
Fragebogen	Hierbei füllen die Beteiligten einen Fragebogen aus. Dieser kann mit Multiple- Choice-Fragen und offene Fragen enthalten. **Vorteile:** • gleichzeitige Informationsgewinnung innerhalb eines großen Personenkreises • weniger zeitaufwendig als ein Interview oder eine Beobachtung • keine zusätzliche Dokumentation der Antworten erforderlich • Möglichkeit der automatischen Auswertung der Antworten **Nachteile:** • höherer Vorbereitungsaufwand als beim Interview • der Rücklauf der Fragebögen ist manchmal unzureichend • Unklarheiten lassen sich schwer klären
Beobachtung	Hierbei werden vor Ort direkt Prozesse begleitet und analysiert. **Vorteile:** • Prozesse sind Objektiv zu beobachten • sehr effektive Erfassung möglich • besonders für die Erfassung zeitlicher und quantitativer Informationen über den Prozess geeignet **Nachteile:** • Zeitlicher Aufwand sehr hoch • es können teilweise nur kleine Ausschnitte eines Prozesses beobachtet werden • Beobachtungen werden i.d.R. aus unangenehm empfunden

Bei allen diesen Methoden müssen verschiedene, auf den Prozess abgestimmte Fragen beantwortet werden. Diese werden im Vorfeld gründlich zusammengestellt. Nachfolgend eine Übersicht an Fragen, welche im Rahmen eines Interviews, eines Fragebogens oder einer Beobachtung eine Rolle spielen können.

Beispielfragestellungen

- Um was für einen Prozess handelt es sich?
- Ist es ein Hauptprozess oder ein Unterprozess?
- Welche Tätigkeiten werden im Prozess in welcher Reihenfolge ausgeführt?
- Welche Prozessschnittstellen gibt es?
- Welche Qualitätsanforderungen sind für einen optimalen Prozessablauf erforderlich?
- Welche Daten werden wann, von wo, wie oft für den Prozess benötigt?
- Wer fällt, welche Entscheidungen in dem Prozess?

Zusätzlich gibt es noch andere Varianten zur Erfassung der notwendigen Informationen, z.B. Brainstrorming, das Sichten von Dokumenten oder Simulationen, welche teilweise im Rahmen von Workshops durchgeführt werden. In der Regel sollten verschiedene Methoden zum Einsatz kommen. Einen idealen Weg gibt es dabei aber nicht. Vielmehr sollte die Form der Analyse dem Kunden angepasst werden, um optimalen Ergebnisse zu erzielen.

✓ Kompetenzcheck

1. Geben Sie an, welche Aussagen richtig sind.

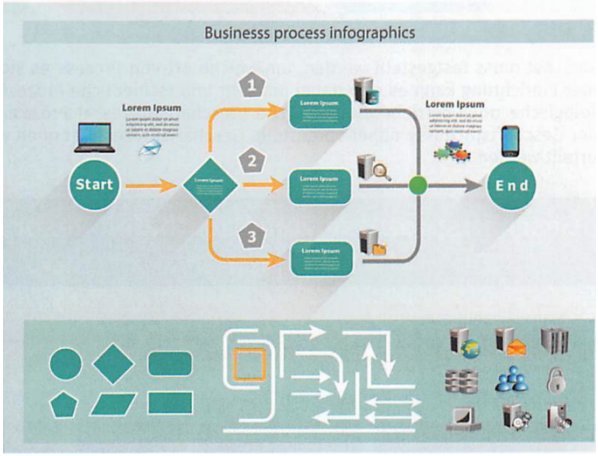
   a. Eine Anforderungsspezifikation wird grundsätzlich immer nur einmal durchgeführt, um spätere Verwirrungen zu vermeiden.
   b. Die Ist-Analyse ist ein wichtiger Bestandteil der Anforderungsspezifikation.
   c. Bei der Ist-Analyse wird u.a. die Verarbeitung der am Prozess beteiligten Daten erfasst.
   d. Die erfassten Anforderungen an die Software werden in funktionale und nicht funktionale Kriterien unterteilt.
   e. Das Ergebnis der Anforderungsanalyse ist in jedem Fall ein Lastenheft.
   f. Interviews und Fragebögen sind Möglichkeiten, um die notwendigen Informationen bei der Anforderungs-spezifikation zu erfassen.

2. Bearbeiten Sie im Handout Kapitel 1 die Aufgaben 1 und 2.

1.2. Arbeitsabläufe und Geschäftsprozesse im Rahmen einer Ist-Analyse beschreiben

➢ **Sie sollen Arbeitsabläufe und Geschäftsprozesse analysieren und modellhaft darstellen.**

Der Zweck der Entwicklung von Software besteht u.a. darin, Arbeitsabläufe oder Prozesse in Unternehmen zu automatisieren. Dazu müssen diese gewisse Voraussetzungen erfüllen: Zum einen müssen die Abläufe klar beschreibbar sein und zum anderen müssen sie sich in der gleichen Form wiederholen. Der letzte Punkt ist Voraussetzung dafür, dass sich die Automatisierung überhaupt lohnt. Laufen Prozesse nur einmal oder selten ab, kann im normalen wirtschaftlichen Umfeld die Steuerung durch den Menschen kostengünstiger sein. Um festzustellen, welche Prozesse mithilfe von Softwareanwendungen ganz oder teilautomatisiert werden können, müssen diese und ihre Schnittstellen im Rahmen von Softwareprojekten zunächst analysiert werden. Daraus werden dann die Anforderungen an die zu entwickelnde Software abgeleitet. Dieses erfolgt im Rahmen einer Ist-Analyse (siehe auch S. 11). Dabei werden die einzelnen Aspekte eines Prozesses analysiert und erfasst.



Grundlegende Begriffe

Prozessanalyse
Eine Prozessanalyse ist eine systematische Untersuchung von Prozessen, um eventuelle Schwachstellen zu erkennen und Verbesserungspotenziale aufzudecken.

Prozess
Unter einem Prozess versteht man die Gesamtheit aller aufeinander einwirkenden Vorgänge innerhalb eines Systems.

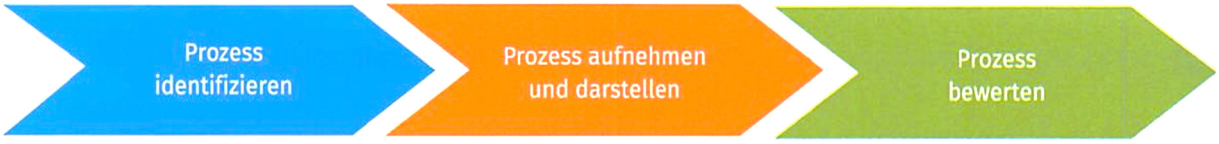
Geschäftsprozess
Ein Geschäftsprozess ist eine sich wiederholende Abfolge von Tätigkeiten die durch einen Input gestartet werden und welche zu einem wirtschaftlich verwertbaren Ergebnis führen.

Prozessschnittstelle
Eine Prozessschnittstelle ist ein Punkt in einem Prozess, an dem der Prozess mit anderen Prozessen in Verbindung tritt. Typische Prozessschnittstellen sind:

- Eingang von Informationen oder Materialien am Anfang oder im Verlaufe des Prozesses
- Austausch von Informationen oder Materialen während des Prozesses
- Ausgang des Prozessergebnisses in Form von Informationen oder Material am Ende des Prozesses

Informationsfluss (Datenfluss)
Der Informationsfluss beschreibt den Weg, welchen mündliche, schriftliche, visuelle oder auditive Daten nehmen, um von einer Quelle zu den Empfängern zu gelangen. In der Informationstechnik spricht man in diesem Zusammenhang auch von Datenfluss.

Eine Prozessanalyse teilt sich in folgende Schritte auf.



Zunächst muss festgestellt werden, um welche Art von Prozess es sich handelt. Je nach Unternehmen oder Einrichtung kann es sich dabei um sehr unterschiedliche Prozesse, z.B. technische, politische, soziologische oder betriebswirtschaftliche (Geschäftsprozesse) Prozesse handeln. Exemplarisch werden hier Geschäftsprozesse näher vorgestellt. Geschäftsprozesse können wiederum in weitere Prozesse unterteilt werden.

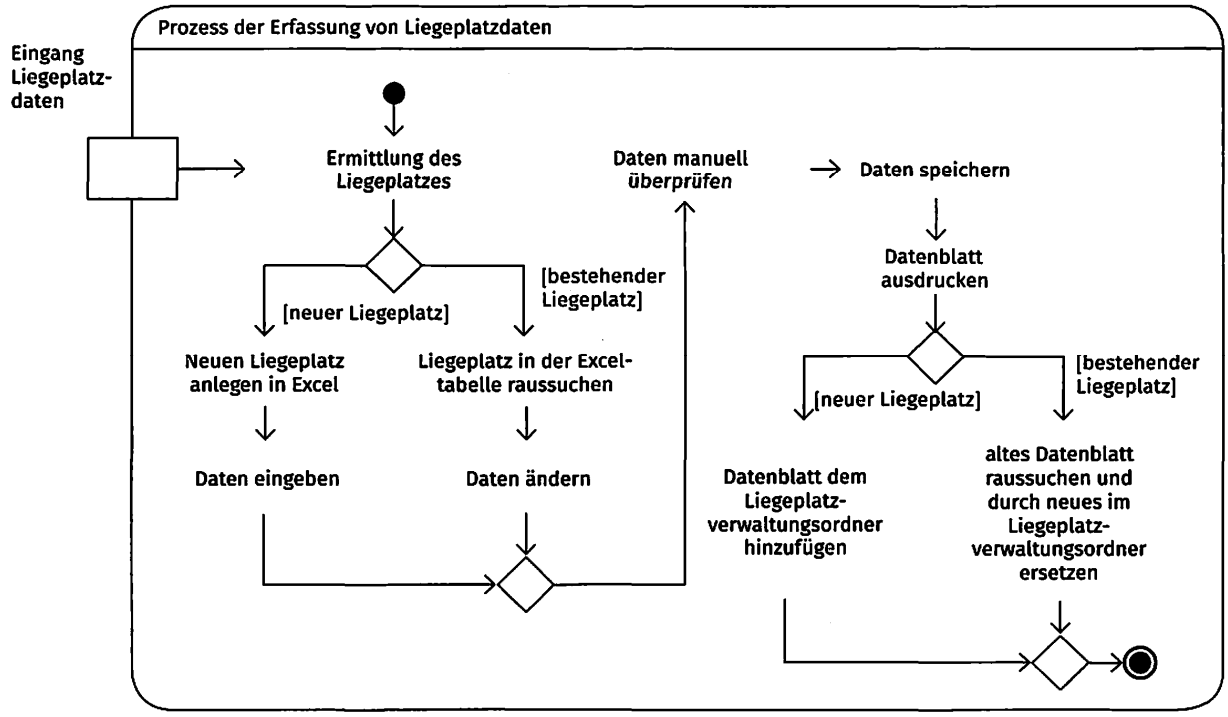


Auch diese Prozesse bestehen wieder aus verschiedenen Teilprozessen. Da alle diese Teilprozesse wiederum von verschiedenen Abteilungen oder Funktionseinheiten ausgeführt werden können, ist es mitunter sehr schwer, einen Überblick über den Gesamtprozess zu bekommen.

Sobald die Art des Prozesses ermittelt wurde, wird mit der Aufnahme des Prozesses begonnen. Dazu wird der Prozess erst einmal in seine einzelnen Teilschritte zerlegt und grafisch dargestellt. Ein wichtiger Punkt dabei ist auch die Aufnahme und Darstellung der Prozessschnittstellen und des Informationsflusses.

Die Darstellung der Prozesse erfolgt in der Regel mithilfe eines Business Process Model (BPM), von einem E3- Value-Model oder von ereignisgesteuerten Prozessketten (EPK). Eine weitere Möglichkeit besteht in der Darstellung in Form einer Wertschöpfungskette. Aber auch ein UML-Aktivitätsdiagramm (siehe auch Kapitel 2.1.2) oder schlicht ein PAP (siehe auch Jahrgangsband 1, Modul 5) eignen sich dazu. Durch eine anschauliche Darstellung werden die Prozesse für alle am Softwareprojekt Beteiligten nachvollziehbar.

Nachfolgend soll im Rahmen des Projektes zur Erstellung einer Software für das „Yachthafen Resort" der Prozess der Erfassung der Liegeplatzdaten untersucht und optimiert werden. Dieser Prozess ist ein Teilprozess des Kernprozesses „Liegeplatzvermietung" Dazu wurde mit den am Prozess beteiligten Personen ein Interview geführt und die daraus gewonnenen Informationen über den bisherigen Prozessablauf mithilfe eines UML-Aktivitätsdiagramms dargestellt.



Nachdem der Prozess erfasst wurde, wird dieser bewertet und analysiert. Hier geht es vor allem darum, Schwachstellen zu erkennen und Verbesserungspotenzial aufzudecken. Des Weiteren wird dabei bestimmt, welche Teile des Prozesses automatisiert werden können. In diesem Fall werden, wie im UML- Diagramm ersichtlich ist, die Daten mithilfe von Excel erfasst und verwaltet. Gleichzeitig werden die Daten auch noch in gedruckter Form in einem entsprechenden Ordner abgelegt. Dieser Prozess ist relativ langsam und fehleranfällig. Durch die redundante Verwaltung der Daten kann es beispielsweise zu Dateninkonsistenzen kommen. Im Rahmen des Softwareprojektes für das „Yachthafen Resort" soll u.a. dieser Prozess optimiert und teilweise automatisiert werden, z. B. die Überprüfung der eingegebenen Daten. Dieses und alle anderen Ergebnisse der Analyse des jeweiligen Prozesses münden in der Formulierung der Anforderungen an das zukünftige Softwareprodukt.

✓ Kompetenzcheck

1. Geben Sie an, welche Aussagen richtig sind.
   a. Die Analyse von Prozessen und deren Schnittstellen ist eine wesentliche Voraussetzung für die Bestimmung der Anforderungen an das zukünftige Softwareprodukt.
   b. Geschäftsprozesse können in Management-, Kern- und Unterstützungsprozesse eingeteilt werden.
   c. Die Montage von Autoteilen gehört bei einem Autohersteller zu den Unterstützungsprozessen.
   d. Der Prozess der Qualitätssicherung gehört bei einem Autohersteller zu den Managementprozessen.
   e. Der Vertrieb der Autos gehört zu den Kernprozessen eines Autoherstellers.
   f. Eine Hardware- oder Softwareschnittstelle kann eine Prozessschnittstelle sein.
   g. Prozesse können mithilfe von BPMN (Business Process Model and Notation) dargestellt werden.
2. Entwickeln Sie in Partnerarbeit Ideen, wie der Prozess der Erfassung der Liegeplatzdaten im Rahmen des Gesamtprojektes optimiert und automatisiert werden kann. Diskutieren Sie Ihre Ideen im Klassenverband.
3. Führen Sie für einen Prozess eine Ist-Analyse durch und stellen Sie diesen grafisch dar. Bearbeiten Sie dazu im Handout die Aufgaben 3, 4 und 5 des Kapitels 1.

1.3. Anforderungskatalog erstellen

➢ **Sie sollen auf der Grundlage der Ist-Analyse die Anforderungen an eine Benutzerschnittstelle formulieren.**



Aus den Ergebnissen der Ist-Analyse werden konkrete Anforderungen an das Softwareprodukt definiert und ausformuliert. Diese werden im Anschluss daran noch einmal dahingehend überprüft, ob sie überhaupt notwendig sind und sie technisch und zeitlich zu realisieren sind.

Die verbliebenen Anforderungen werden sodann Softwarekriterien zugeordnet und in funktionale und nicht funktionale Anforderungen klassifiziert (siehe auch Jahrgangsband 1, Modul 5).

Funktionale Anforderungen beziehen sich darauf, was das System leisten soll. Diese sind von Software zu Software sehr unterschiedlich und spezifisch. Bei den funktionalen Anforderungen werden die benötigten Eingabedaten, die erforderlichen Verarbeitungsschritte und die erwarteten Ausgabendaten genau beschrieben. Nicht funktionale Anforderungen beziehen sich darauf, wie das System seine Leistungen erbringen soll. Sie werden in der Anforderungsspezifikation genau wie die funktionalen Anforderungen beschrieben. Die nicht funktionalen Anforderungen werden zudem in Qualitätsanforderungen und Randbedingungen unterteilt. Die Qualitätsanforderungen richten sich dabei meistens an Softwarequalitätskriterien wie z. B. Effizienz oder Wartbarkeit aus, während die Randbedingungen technologische, organisatorische und normative Anforderungen beschreiben.

Nach der Einteilung der Anforderungen in verschiedene Kategorien sollten diese zum Schluss noch nach Wichtigkeit priorisiert werden, um dem Kunden zügig erste Ergebnisse vorstellen zu können. Wie die einzelnen Anforderungen beschrieben werden und wie der Anforderungskatalog später aussieht, hängt viel von dem verwendeten Vorgehensmodell ab. So werden beispielsweise bei Scrum (siehe auch Jahrgangsband 2, Modul 8) zur Beschreibung der Anforderungen Epics und User Stories verwendet. Diese werden immer aus der Perspektive des Benutzers geschrieben, um ein besseres Verständnis für die Belange des Benutzers zu entwickeln. Es wird zunächst mit der Beschreibung der Epics begonnen. Epics geben grob die wichtigsten Anforderungen an die Software wieder. Diese werden dann später in einzelne User Stories aufgeteilt, welche mindestens aus ein oder zwei Sätzen bestehen. Im Anforderungskatalog, z.B. dem Product Backlog bei Scrum, werden alle Anforderungen an die zukünftige Software zusammengefasst. Da es in diesem Modul schwerpunktmäßig um die Entwicklung von Benutzerschnittstellen für Software geht, werden nachfolgend nur die Anforderungen an diese näher betrachtet.

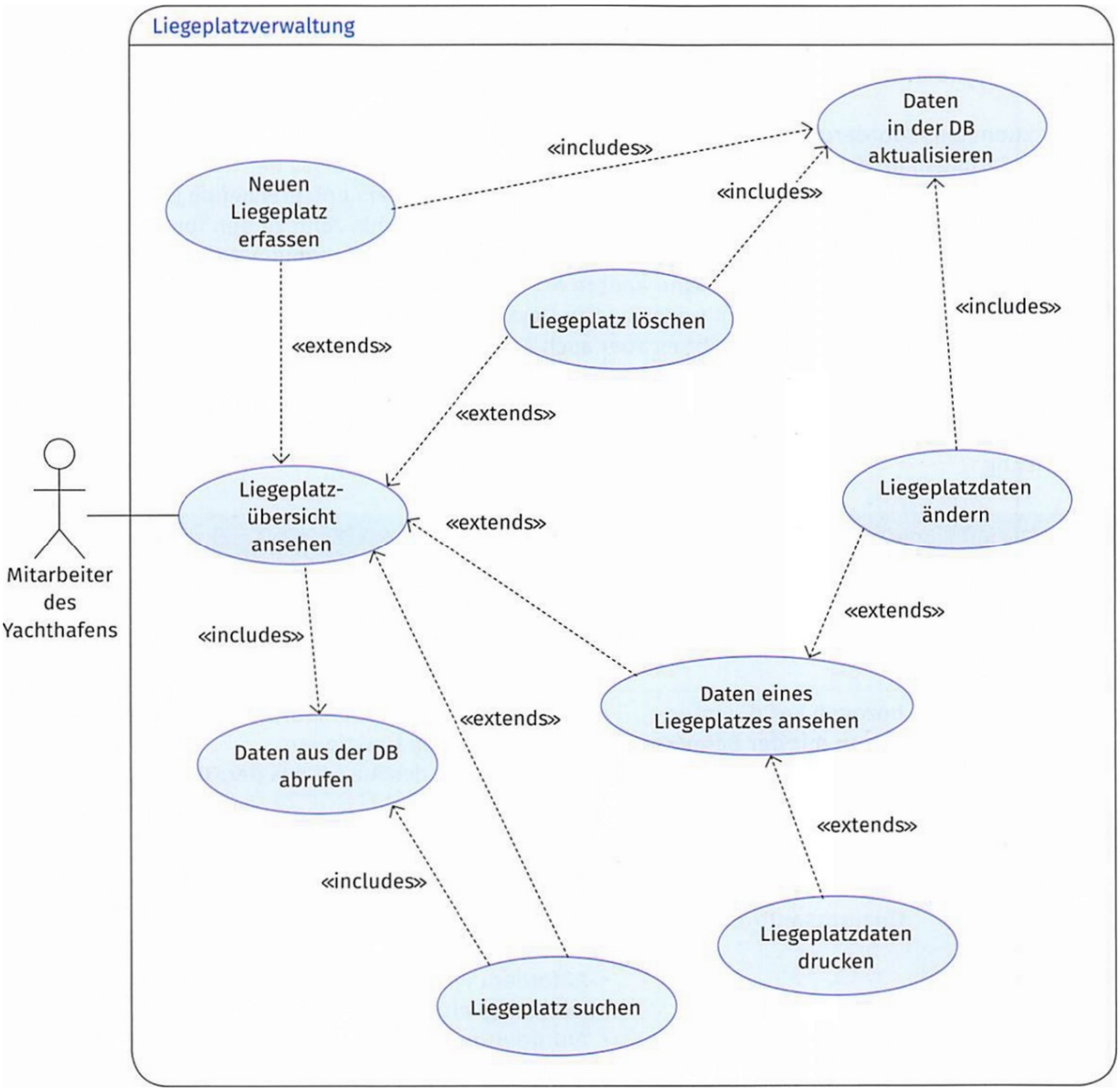
1.3.1. Funktionale und nicht funktionale Anforderungen an Benutzerschnittstellen

Die funktionalen Anforderungen an eine Benutzerschnittstelle ergeben sich zunächst aus den funktionalen Anforderungen an die Anwendung. Durch die Benutzerschnittstelle muss der Nutzer in die Lage versetzt werden, mit der Anwendung zu interagieren, um z. B. dem System entsprechende Eingabedaten zur Verfügung zu stellen, welche dann weiterverarbeitet werden. Weiterhin zählt zu den funktionalen Anforderungen, welche Endgeräte zum Einsatz kommen sollen. Oftmals sind mobile Versionen der Benutzerschnittstelle notwendig, welche auch eine andere Art der Interaktion erfordern. Bei mobilen Versionen kann sich die Benutzeroberfläche im Gegensatz zu Desktopanwendungen erheblich unterscheiden. Neben der „fachlichen" Funktionalität gibt es aber auch Anforderungen an eine Funktionalität, welche rein auf die Interaktion des Benutzers mit der Oberfläche ausgerichtet ist, z. B. Scroll- oder Zoomfunktionen. Im Bereich der nicht funktionalen Anforderungen spielt bei den Qualitätsanforderungen das Qualitätskriterium „Benutzbarkeit" die größte Rolle. In diesem Bereich sind aber auch Zuverlässigkeit und Effizienz sehr wichtig.

Funktionale und nicht funktionale Anforderungen an eine Benutzerschnittstelle		
Funktionale Anforderungen	fachlich; bezogen auf den Arbeitsablauf oder Geschäftsprozess	• Eingabe und Löschfunktionalität für notwendige Daten des Prozesses • Anzeigen von Grafiken und Bildern mit Inhalten des jeweiligen Prozesses
	bezogen auf die Interaktion mit der Benutzerschnittstelle	• Bedienbarkeit, z. B. mit Maus, Tastatur, Sprache • Scroll- und Zoomfunktionen • Anpassbarkeit des Aussehens der Oberfläche
	bezogen auf geplante Endgeräte und Softwareumgebungen	• Tablets, Smartphone, Desktopanwendung, stationäre Touch Panel für Automaten • Betriebssysteme, Browser
Nicht funktionale Anforderungen	Qualitätsanforderungen	• Benutzbarkeit: • Anforderungen an die Selbstbeschreibbarkeit • Anforderungen an die Erwartungskonformität • Zuverlässigkeit: • Anforderungen an die Fehlertoleranz der Benutzerschnittstelle • Effizienz: • Anforderungen an die Darstellung, z. B. von 3-D-Animationen • Anforderungen an der Ressourcenverbrauch, z. B. bei speziellen Endgeräten
	technische, organisatorische und normative Rahmenbedingungen	• vorgegebene Betriebssysteme und Browser • Rechtvergabe für bestimmte Funktionalität • Grad der Barrierefreiheit oder Mehrsprachigkeit

1.3.2. Anforderungskatalog für das Beispielszenario

An dieser Stelle sollen nun exemplarisch die Anforderungen an die Benutzerschnittstelle für die Liegeplatzverwaltung bestimmt werden. Als Grundlage dafür dient das abgebildete Anwendungsfalldiagramm (siehe auch Kapitel 2.1.2 oder Jahrgangsband 2, Modul 8), welches die Grundfunktionalität der Liegeplatzverwaltung beschreibt.



Da ein Anwendungsfalldiagramm auch immer die Interaktion eines oder mehrerer Benutzer mit dem System abbildet, können hieraus unmittelbar Anforderungen an die Benutzerschnittstelle abgeleitet werden. In diesem Fall kann daraus abgeleitet werden, dass die Benutzerschnittstelle u.a. Möglichkeiten zum Hinzufügen und Löschen von Liegeplätzen bereitstellen muss.

In der folgenden Vorlage hat eine Auszubildende der JIKU IT-Solutions die Anforderungen für die Benutzerschnittstelle für die Liegeplatzverwaltung erfasst.

Anforderungen an die Benutzerschnittstelle
Liegeplatzverwaltung

JIKU IT-Solutions

Kriterien	Anforderungen
Funktionale Anforderungen (prozessbezogen)	*Die Benutzeroberfläche soll Nutzern folgende Funktionalität bieten:* • *alle Liegeplätze in einer zentralen Übersicht ansehen,* • *bestimmte Liegeplätze suchen,* • *Liegeplätze nach bestimmten Kriterien filtern,* • *neue Liegeplätze erfassen,* • *bestehende Liegeplätze löschen* • *Daten bestehender Liegeplätze zu ändern.*
Funktionale Anforderungen (interaktionsbezogen)	*Es soll eine grafische Benutzeroberfläche (GUI) entwickelt werden, welche mit der Maus und der Tastatur bedient werden kann.*
Funktionale Anforderungen (technisch)	*Es soll zunächst nur eine Desktop-Oberfläche für stationäre Computersysteme entwickelt werden (Betriebssystem: Windows).*
Nicht funktionale Anforderungen (Qualitätsanforderungen)	*Die Oberfläche soll auf der Grundlage der Interaktionsprinzipien ISO-Norm 9241-110 gestaltet werden. Außerdem ist auf ein gutes UX Design zu achten.*
Nicht funktionale Anforderungen (Rahmenbedingungen)	*Es sind die Vorgaben zur Corporate Identity des „Yachthafen Resorts" zu beachten und umzusetzen, z. B. die Verwendung des Logos des Unternehmens.*

✓ Kompetenzcheck 0

1. Geben Sie an, welche Aussagen richtig sind.

   a. Angaben zur Wartbarkeit einer Software gehören zu den funktionalen Anforderungen.
   b. Anforderungen an die Fehlertoleranz einer Software zählen zu den nicht funktionalen Anforderungen.
   c. Rahmenbedingungen, z.B. den Einsatz der Software im deutschsprachigen und nicht deutschsprachigem Raum kann eine mehrsprachige Gestaltung der Benutzerschnittstelle notwendig machen.
   d. Wenn die Benutzerschnittstelle 3-D-Animationen aufweisen soll, können Fragen der Effizienz eine große Rolle spielen.
   e. Betriebssysteme und Browser spielen bei den Anforderungen an eine Benutzerschnittstelle überhaupt keine Rolle.
   f. Die Sprachausgabe von Inhalten gehört zu interaktionsbezogen Funktionalität von Benutzerschnittstellen.
   g. Vorgaben für Farben oder Schriftarten müssen nicht erfasst werden, da diese für die Gestaltung der Benutzerschnittstelle nicht von Bedeutung sind.

2. Finden Sie in Gruppenarbeit auf Grundlage der hier vorgestellten Beispiele weitere funktionale und nicht funktionalen Anforderungen für Benutzerschnittstellen. Diskutieren Sie Ihre Ergebnisse im Klassenverband.

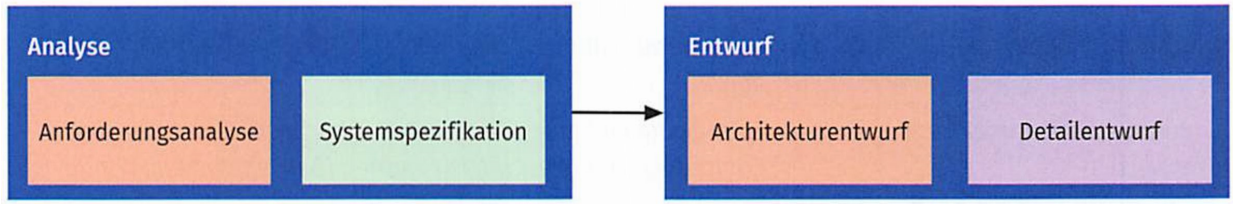
3. Bearbeiten Sie im Handout die Aufgabe 6 des Kapitels 1 und überprüfen Sie im Rahmen der Prüfungsvorbereitung Ihr Wissen bezüglich der Anforderungs- und Prozessanalyse mithilfe der Aufgabe 7.

2. Ein Softwareprodukt designen

➢ **Sie sollen Ihr Basiswissen über das Planen und Entwerfen von Software vertiefen und erweitern.**

Das Designen (Entwerfen) einer Software ist der Teil des Softwareentwicklungsprozesses (siehe Jahrgangsband 1, Modul 5), bei dem die Architektur, die Komponenten, die Schnittstellen und andere Eigenschaften wie die Datenstrukturen und Algorithmen einer Software definiert und geplant werden. Der Entwurf der Software ist eine sehr wichtige Phase, da die gesamte Anwendung auf den Entscheidungen, welche hier getroffen werden, aufbaut.





Mit dem Entwurf wird begonnen, nachdem die grundlegenden Anforderungen an die Software feststehen. Diese müssen dabei noch nicht vollständig sein, wie es vor allem bei agilen Vorgehensmodellen der Fall ist. Abhängig vom gewählten Vorgehensmodell (siehe Jahrgangsband 1, Modul 5 und Jahrgangsband 2, Modul 8) ist der Softwareentwurf eine in sich abgeschlossene Phase, z. B. beim Wasserfallmodell, oder wird wiederholt durchlaufen werden, z. B. bei Scrum. Dabei gibt es mehrere Entwurfsebenen, welche fließend ineinander übergehen und deren Ergebnisse während der Entwicklungszeit angepasst, verfeinert und optimiert werden.

Entwurfsebenen beim Softwaredesign

Architekturentwurf
Das architektonische Design stellt die höchste abstrakte Version des Systems dar. Es identifiziert die Software als ein System, bei dem viele Komponenten miteinander interagieren. Dabei geht es noch nicht um einen detaillierten Entwurf, sondern es werden die Zusammenhänge zwischen den Anforderungen und dem zu konstruierenden System zunächst beschrieben und begründet. In einem ersten Schritt wird der grundlegende Aufbau des Softwaresystems festgelegt. Dieser wird dann in einem zweiten Schritt in einzelne Subsysteme und Module weiter unterteilt und deren Beziehungen untereinander beschrieben. Das Ergebnis dieses Prozesses ist die sogenannte Softwarearchitektur. Die Softwarearchitektur wird neben den funktionalen Anforderungen wesentlich durch die Anforderungen an Softwarequalitätskriterien wie beispielsweise Wartbarkeit, Sicherheit oder Performance bestimmt.

Detailentwurf
Der Detailentwurf beschreibt den genauen Aufbau der Module und deren Implementierungen. Hier werden die logische Struktur jedes Moduls, seine Komponenten und seine Schnittstellen definiert. Außerdem werden die Datenstruktur und die Algorithmen der verschiedenen Module entworfen. Das Ergebnis der detaillierten Entwurfsphase wird in der Regel als Modulentwurf bezeichnet.

Nachfolgend wird die Architektur einer Software näher betrachtet. Der Detailentwurf wird genauer in Kapitel 2.1.2 erläutert.

2.1. Benutzerschnittstellen in den Kontext der Softwarearchitektur einordnen

➢ **Sie sollen sich die Grundlagen über die Architektur von Software erarbeiten.**

Der erste Schritt bei einem Softwareentwurf ist der Architekturentwurf. Beim Architekturentwurf geht es noch nicht um einen detaillierten Entwurf, sondern es werden die Zusammenhänge zwischen den Anforderungen und dem zu konstruierenden System beschrieben und begründet. Je nach Umfang der zu entwickelnden Software kann dabei die Architektur sehr einfach oder auch sehr komplex ausfallen.



Softwarearchitektur
Die Softwarearchitektur stellt den Bauplan eines Softwaresystems dar. Dabei werden die einzelnen Komponenten und deren Verbindungen untereinander beschrieben.

Bei kleinen Softwareprojekten wird die Architektur vom Entwickler selbst bestimmt oder er verwendet vorgefertigte Architekturmuster (siehe auch Kapitel 1.2.2), bei größeren Softwa re Projekten wird diese meistens von erfahrenen Softwarearchitekten entworfen und der Entwickler muss diese im Anschluss umsetzen. Hierbei sind grundlegende Kenntnisse über verschiedenen Softwarearchitekturen von Vorteil. Dieses trifft vor allem auch dann zu, wenn eine bestehende Software erweitert werden soll. In diesem Fall ist ein grundlegendes Verständnis für den architektonischen Aufbau der Software sehr hilfreich, um sich schnell in die bestehende Implementierung hineinzudenken. Eine Softwarearchitektur sollte sorgfältig geplant werden, denn einmal in diesem Bereich getroffene Entscheidungen sind später nur mit hohem Aufwand zu verändern. Eine gute Softwarearchitektur erhöht generell die Qualität der Software in Bereichen wie Wartbarkeit, Effizienz oder Zuverlässigkeit, was am Ende z. B. zu einer hohen User Experience (siehe S. 35) sowie Kundenzufriedenheit und damit zu einer hohen Kundenbindung führt.

Für eine gute Softwarearchitektur sind verschiedene Merkmale entscheidend. Exemplarisch sind einige davon in der nachfolgenden Tabelle aufgeführt.

Wesentliche Merkmale einer guten Softwarearchitektur

Modularität
Die Aufteilung eines großen Softwareprojekts in kleinere Teile wird als Modularität bezeichnet. Die Modularität ist eine Grundvoraussetzung für ein skalierbares und wartbares Softwaredesign. Dadurch, dass das Projekt in verschiedene Komponenten unterteilt ist, ist es möglich, diese parallel zu entwickeln. Durch die Modularität wird auch das Testen vereinfacht.

Entkopplung
Die Kopplung bezieht sich auf das Ausmaß der wechselseitigen Abhängigkeit (Interdependenz) zwischen den einzelnen Softwaremodulen und wie eng diese miteinander verbunden sind. Eine niedrige Kopplung ist ein Merkmal eines guten Designs. Bei geringer Kopplung können Änderungen in jedem Modul einzeln vorgenommen werden, ohne dass die anderen Module davon beeinflusst werden.

Abstraktion
Der Prozess der Identifizierung des wesentlichen Verhaltens der Module durch das Trennen von der Implementierung und dem Entfernen irrelevanter Details wird als Abstraktion bezeichnet. Eine hohe Abstraktion trägt entscheidend zu einer geringen Kopplung bei.

Einfachheit
Das Ziel eines guten Softwaredesigns ist Einfachheit. Jede Aufgabe hat ein eigenes Modul, das eigenständig genutzt und modifiziert werden kann. Dieses hilft den Quellcode wartbar zu halten und minimiert die Anzahl der potentiellen Fehler.

Vollständigkeit
Ein gutes Softwaredesign stellt die Hinlänglichkeit und Vollständigkeit der Software in Bezug auf die festgelegten Anforderungen sicher.

Parallelität
Unter Parallelität wird die Möglichkeit der Software verstanden, mehrere Aufgaben gleichzeitig auszuführen. Im Softwaredesign wird Parallelität implementiert, indem die Software in mehrere unabhängige Ausführungseinheiten aufgeteilt und diese dann parallel ausgeführt werden. Damit wird die Effizienz der Software gesteigert.

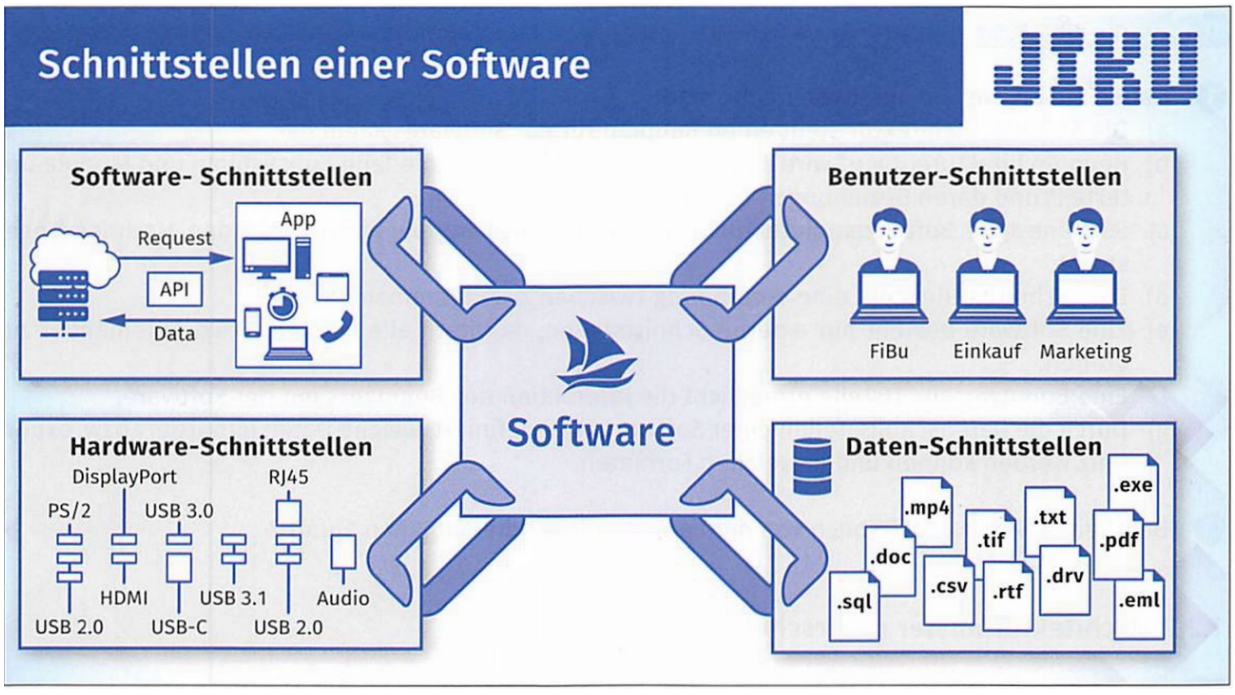
Ein wesentlicher Bestandteil der Softwarearchitektur sind die Module. Diese werden im Zuge der Modularisierung eines Softwaresystems (siehe auch Kapitel 2.1) entwickelt. Die Beschreibung der Architektur des Softwaresystems, dessen Module und deren Beziehungen untereinander kann u.a. mithilfe von UML- Komponentendiagrammen und UML-Verteilungsdiagrammen erfolgen (siehe auch Jahrgangsbuch 2, Modul 8).

Modul
Ein Modul ist ein Baustein eines Softwaresystems, welcher eine funktional geschlossene Einheit darstellt und einen bestimmten Dienst bereitstellt, welcher über eindeutig spezifizierte Schnittstellen genutzt werden kann.

Ein weiterer Bestandteil der Softwarearchitektur ist neben der Beschreibung der Softwaremodule die Beschreibung der Schnittstellen der Software. Dabei werden interne und externe Schnittstellen unterschieden. Die internen Schnittstellen der einzelnen Module dienen zur Interaktion mit anderen Modulen der Software.

Schnittstelle
Eine Schnittstelle ist eine Verbindung zwischen zwei Systemen, die eine Kommunikation bzw. eine Übertragung ermöglicht.

Die externen Schnittstellen ermöglichen die Kommunikation mit der Software. Diesen kommt eine besonders große Bedeutung zu. Über sie können z. B. heterogene Systeme miteinander verbunden und so Insellösungen vermieden werden. Zudem ermöglichen sie eine unkomplizierte Interaktion der Benutzer mit der Software. Es werden Daten-, Hardware-, Software- und Benutzerschnittstellen unterschieden.



Unterscheidung von Schnittstellen einer Software

Benutzerschnittstelle
Eine Benutzerschnittstelle stellt eine Schnittstelle zwischen der Software und dem Benutzer der Software dar. Sie nimmt die Eingaben des Benutzers entgegen, leitet diese zur Verarbeitung weiter und stellt dem Benutzer danach die Ergebnisse wieder zur Verfügung. Bei den Benutzerschnittstellen werden noch verschiedene Formen unterschieden, z. B. CLI, GUI (siehe Kapitel 1.3.1).

Softwareschnittstelle
Softwareschnittstellen bilden den Übergang zwischen unterschiedlichen Programmen und ermöglichen dadurch den Datenaustausch zwischen diesen. Zu den Softwareschnittstellen werden vor allem API (Application Programming Interface) und ABI (Application Binary Interface) gezählt. Durch eine API kann eine Software in ein anderes System eingebunden werden. Damit ein Programm seine Funktionalität anderen Programmen zur Verfügung stellen kann, muss eine entsprechende Programmierschnittstelle implementiert werden.

Hardwareschnittstelle
Die Hardwareschnittstelle ist eine Schnittstelle zwischen physikalischen Systemen in der Elektrotechnik und Elektronik. Damit eine Software mit den erforderlichen Hardwareschnittstellen, z. B. USB-Anschlüssen, interagieren kann, muss eine entsprechende Funktionalität im Programm implementiert werden (siehe auch Jahrgangsband 1, Modul 2 und Modul 3).

Datenschnittstelle
Durch Datenschnittstellen wird festgelegt, welche Daten bzw. Datenformate die Software einlesen oder ausgeben kann. Datenschnittstellen dienen vor allem dazu, größere Datenmengen automatisiert dem Programm zur Verfügung zu stellen oder dieses aus dem Programm auszulesen.

Während früher dem Entwurf und dem Design der Benutzerschnittstelle nicht so viel Aufmerksamkeit geschenkt wurde, gewinnt dieser Bereich immer mehr an Bedeutung, sodass er heutzutage in vielen Fällen schon einen eigenen umfassenden Entwicklungsschritt im Softwarenentwicklungszyklus darstellt.

✓ Kompetenzcheck

1. Geben Sie an, welche Aussagen richtig sind.

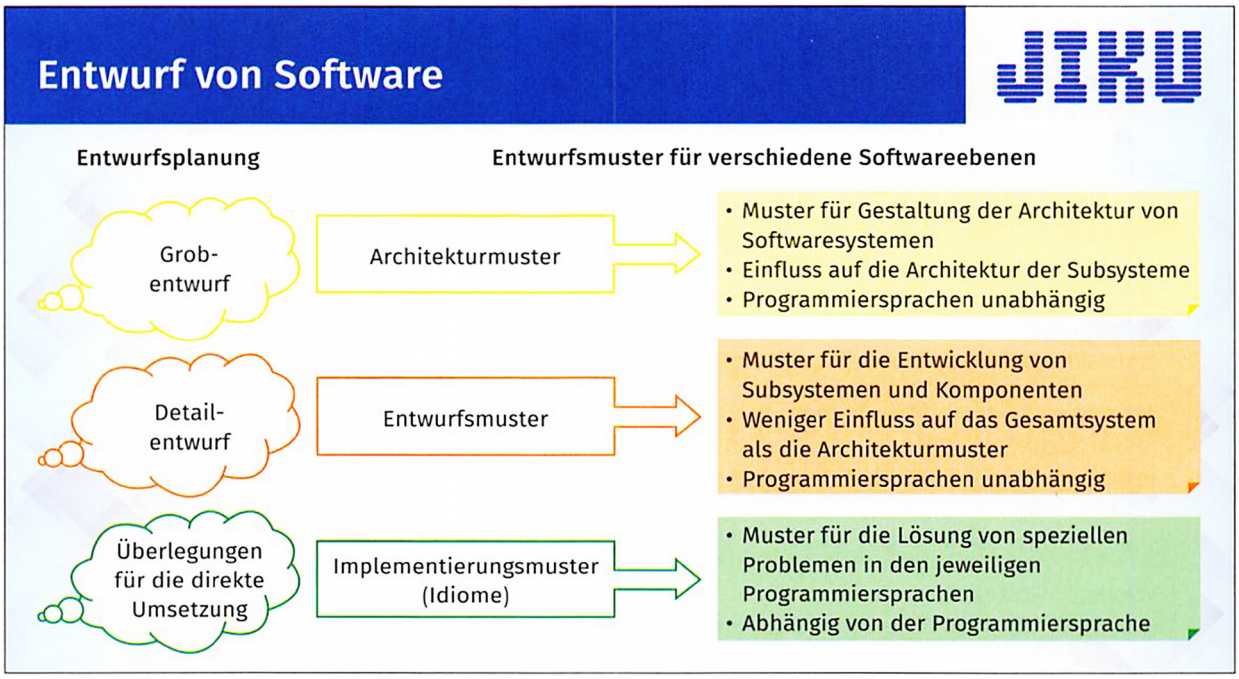
   a. Die Softwarearchitektur stellt einen Bauplan für ein Softwaresystem dar.
   b. Beim Architekturentwurf wird das Softwaresystem in verschiedene Subsysteme und Module unterteilt und deren Beziehungen untereinander beschrieben.
   c. Für eine gute Softwarearchitektur wird eine niedrige Kopplung zwischen den Modulen angestrebt.
   d. Eine Schnittstelle stellt eine Verbindung zwischen zwei Systemen dar.
   e. Eine Software besteht nur externe Schnittstellen, da intern alle Module direkt aufeinander zugreifen.
   f. Eine Benutzerschnittstelle ermöglicht die Interaktion des Benutzers mit der Software.
   g. Durch die Datenschnittstellen einer Software wird definiert, welche Daten importiert bzw. exportiert werden können und in welchen Formaten.

2. Bearbeiten Sie im Handout Kapitel 2 die Aufgaben 1 und 2.

2.2. Architekturmuster unterscheiden

➢ **Sie sollen sich die Grundlagen über Entwurfsmuster erarbeiten und Architekturmuster unterscheiden und beschreiben können.**

Um wiederkehrende Entwurfsprobleme nicht immer wieder neu lösen zu müssen, wurden sogenannte Entwurfsmuster entwickelt, welche in den verschiedenen Planungsstufen zum Einsatz kommen.



Zunächst kommen die Architekturmuster zum Einsatz. Im Gegensatz zu den Entwurfs- oder Implementierungsmustern wird durch diese die grundlegende Organisation und Interaktion zwischen den Komponenten einer Anwendung beschrieben. Die verschiedenen Architekturmuster lassen sich in vier große Bereiche einordnen:

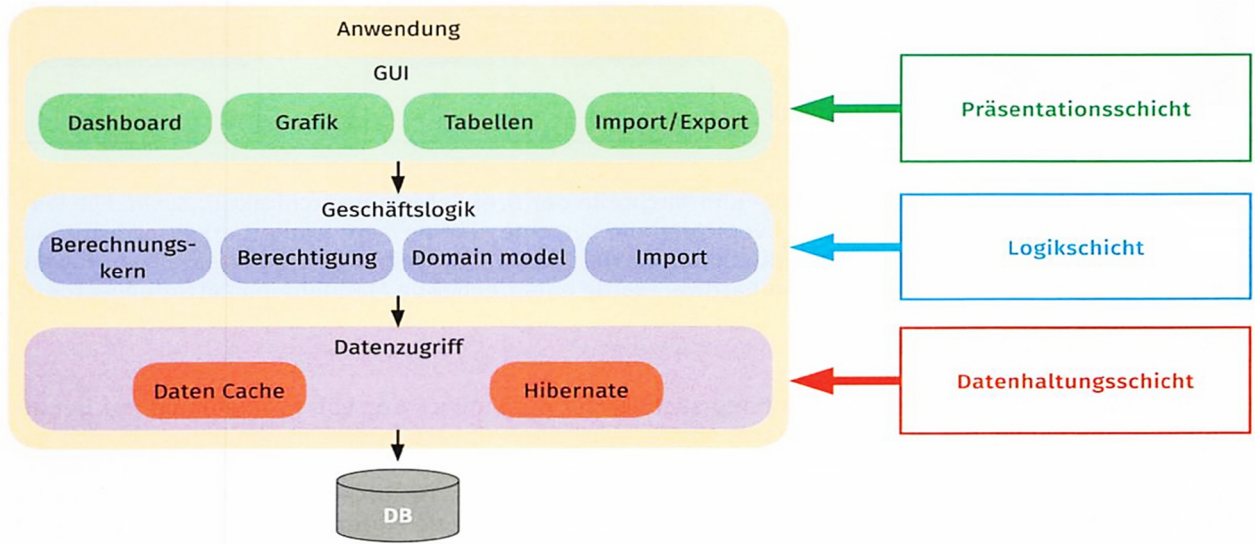
Kategorien von Architekturmustern	
Mud-to-Structure	Hierunter fallen alle Architekturmuster, die dabei helfen ein System aus zahlreichen Komponenten zu organisieren, z. B. Schichtenarchitektur, Blackboard.
Verteilte Systeme	Dieser Bereich enthält Architekturmuster, die für die Verwendung von Diensten und Ressourcen in Netzwerken gedacht sind, z. B. Client-Server-Architektur, Peer-to-Peer.
Interaktive Systeme	In diesen Bereich fallen alle Architekturmuster, die sich auf die Interaktion eines Softwaresystems mit Menschen spezialisieren, z. B. Model View Controller (MVC), Model View Presenter (MVP).
Adaptive Systeme	Architekturmuster aus diesem Bereich beschäftigen sich vorrangig mit der Erweiterbarkeit und der Anpassungsfähigkeit von Softwaresystemen, z. B. Reflection, Dependency Injection.

Nachfolgend sollen die Drei-Schichten-Architektur, die Client-Server-Architektur und das MVC-Entwurfsmuster beispielhaft näher betrachtet werden.

2.2.1. Drei-Schichten-Architektur

Die Drei-Schichten-Architektur ist eine Spezialform der Schichtenarchitektur. Generell wird bei der Schichtenarchitektur die Software in verschiedenen Schichten (Layer, Tier) aufgeteilt, welche unterschiedliche Aufgaben erfüllen. Bei einer Drei-Schichten-Architektur erfolgt eine Aufteilung in drei Schichten:

Schichten einer Drei-Schichten-Architektur	
Präsentationsschicht (Client Tier, Front End)	Diese Schicht ist für die Repräsentation der Daten, Benutzereingaben und die Benutzerschnittstellen verantwortlich.
Logikschicht (Application-Server Tier, Middle Tier, Enterprise Tier)	Diese Schicht verarbeitet die Daten. Hier ist die Anwendungslogik vereint.
Datenhaltungsschicht (Data-Server Tier, Back End)	Diese Schicht ist verantwortlich für das Speichern und Laden von Daten. Hier ist in der Regel auch die Datenbank angesiedelt.

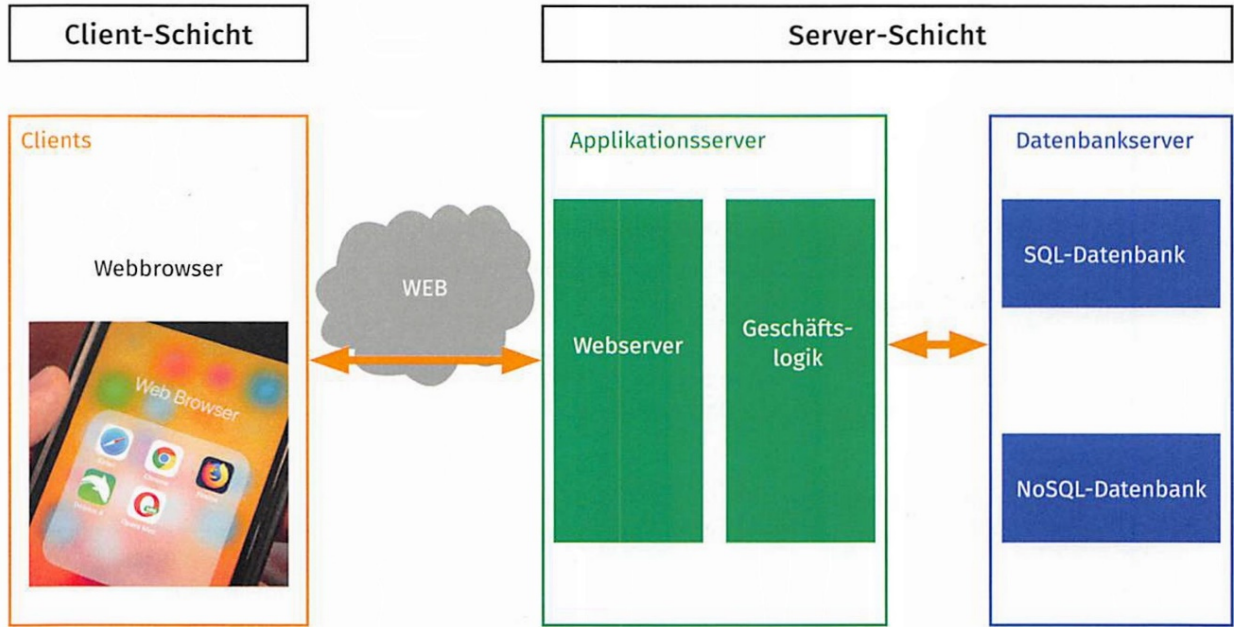


Die Drei-Schichten-Architektur hat verschiedene Vor- und Nachteile:

Vor- und Nachteile der Drei-Schichten-Architektur	
Vorteile	**Nachteile**
• Reduzierung von Abhängigkeiten zwischen den einzelnen Komponenten • Definierte Schnittstellen zwischen den Schichten • Änderungen wirken sich nur selten auf andere Schichten aus • Austausch einer Schicht ist ohne Probleme möglich	• Reduzierung der Ausführungsgeschwindigkeit der Software durch die Weiterleitung und Transformation der Daten über verschiedene Schichten

2.2.2. Client-Server-Architektur

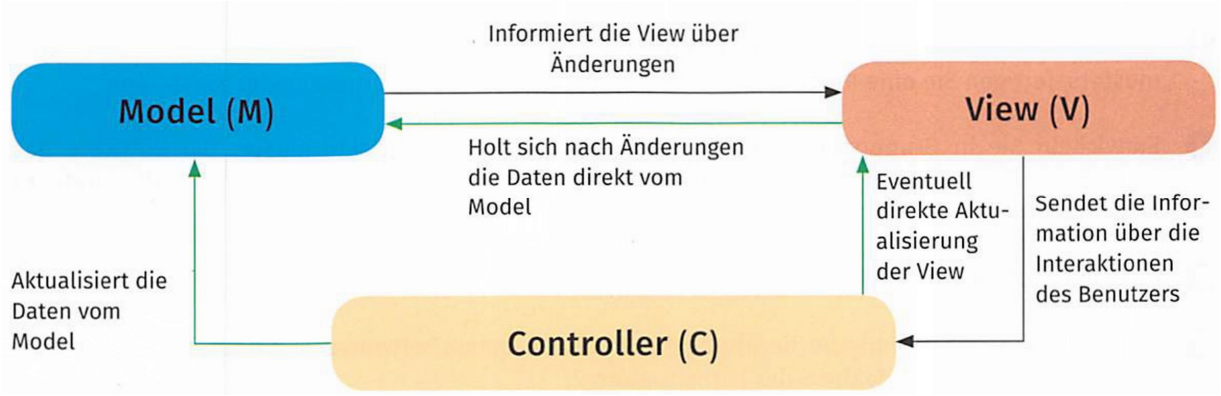
Die Client-Server-Architektur ist ein Architekturmuster für verteilte Systeme. Dabei bieten Subsysteme („Server") bestimmte Dienste an, die von anderen Subsystemen („Clients") genutzt werden. Im nachfolgenden Beispiel wird die Drei-Schichten-Architektur mit der Client-Server-Architektur verknüpft. Dabei werden die verschiedenen Schichten auf unterschiedliche Systeme verteilt.



Zunächst einmal gelten auch hier die Vor- und Nachteile der Drei-Schichten-Architektur. Zusätzlich wird durch die Verteilung der Anwendung auf verschiedene Systeme eine bessere Skalierbarkeit erreicht. Auf der anderen Seite kann beispielsweise durch den Ausfall des Webservers die Funktionalität der Anwendung unterbrochen werden. Hier müssen hardwareseitig entsprechende Maßnahmen getroffen werden.

2.2.3. Model View Controller (MVC)

Das MVC-Muster ist ein Muster, welches hauptsächlich für die Entwicklung von Programmen mit Benutzeroberflächen verwendet wird. Bei diesem Muster wird die Software in drei unabhängige, in sich geschlossene Komponenten unterteilt: das Model, die View und den Controller. Die Komponenten interagieren miteinander, um die Gesamtfunktionalität der Software umzusetzen. Die Idee hinter dem MVC-Muster ist die klare Abgrenzung zwischen den Daten, deren Darstellung und den Interaktionen des Benutzers mit GUI. Mittlerweile ist es eines der am weitesten verbreiteten Muster für die Präsentationsschicht der Drei- Schichten-Architektur von Software.



Rollen und Aufgaben im MVC-Muster

View
Die View zeigt dem Benutzer die Daten, visualisiert die digitalen Informationen in passender Art und Weise, nimmt Benutzerinteraktionen entgegen und leitet diese an den Controller weiter.

Controller
Stellt die Schnittstelle zwischen der View und dem Model dar. Hier ist die Logik der Anwendung zu finden. Die Aufgabe des Controllers ist es, die Benutzerinteraktionen auszuwerten und Veränderungen an den Daten des Models vorzunehmen.

Model
Hier werden die Daten verwaltet. Es beschreibt die Datenstruktur und die Daten. Die Daten werden unabhängig vom Typ und der Art der Datenbank bereitgestellt. Das Model informiert die View über Änderungen bei den Daten.

Vor- und Nachteile des MVC-Musters	
Vorteile	**Nachteile**
• mehrere Ansichten (Views) des Models möglich • austauschen von Views möglich • leichtes Hinzufügen von neuen Views • Synchronisierte Views • unterstützt sehr gut die agile Softwareentwicklung	• erhöht Komplexität der Anwendung

✓ Kompetenzcheck

1. Geben Sie an, welche Aussagen richtig sind.

   a. Die Softwarearchitektur stellt einen Bauplan für Software dar.
   b. Der Softwareentwurf ist immer eine in sich abgeschlossene Phase, welche nur einmal durchlaufen wird.
   c. Idiome bestimmen den generellen Aufbau einer Software.
   d. Interaktive Architekturmuster beziehen sich vorrangig auf die Erweiterbarkeit und die Anpassungsfähigkeit von Softwaresystemen.
   e. Durch die Schichtenarchitektur wird eine Software in unterschiedliche Schichten aufgeteilt, welche über Schnittstellen miteinander kommunizieren.
   f. Die Client-Server-Architektur und die Drei-Schichten-Architektur schließen sich gegenseitig aus.
   g. Ein Vorteil der Drei-Schichten-Architektur ist die hohe Abhängigkeit der einzelnen Komponenten untereinander.
   h. Beim MVC-Muster übernimmt die View in der Regel einen großen Teil der Datenverwaltung.
   i. Ein Vorteil des MVC-Musters ist die Möglichkeit des unkomplizierten Hinzufügens von zusätzlichen Views.

2. Recherchieren Sie in Gruppenarbeit den Aufbau und die Funktionsweise eines weiteren Architekturmusters, fertigen Sie eine Präsentation an und stellen Sie Ihre Ergebnisse in der Klasse vor.

3. Entwickeln Sie in Gruppenarbeit eine Idee, nach welchem Architekturmuster die Software des „Yachthafen Resorts" aufgebaut sein könnte. Präsentieren Sie im Anschluss Ihre Idee den anderen Auszubildenden.

4. Bearbeiten Sie im Handout die Aufgaben 3, 4, 5 des Kapitels 2.

5. Überprüfen Sie zur Prüfungsvorbereitung Ihr Wissen bezüglich Softwarearchitektur und Architekturmustern mithilfe der Aufgabe 6 des Kapitels 2.

3. Benutzerschnittstellen funktionsgerecht und ergonomisch konzipieren

➢ **Sie sollen Benutzerschnittstellen für eine Software entwerfen. Dazu erarbeiten Sie sich die notwendigen Grundlagen.**

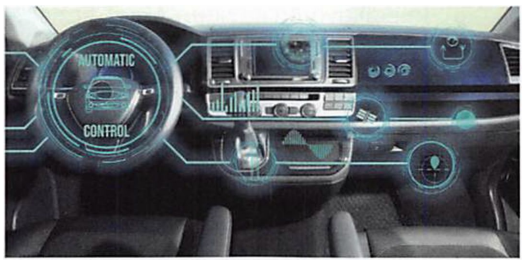


Die Digitalisierung erfasst in allen Bereichen unseren Alltag. Zunehmend können Dienste von Dienstleistern nur noch digital über das Internet in Anspruch genommen werden. Deswegen ist es notwendig, einen nutzergerechten Zugang zu diesen digitalen Diensten zu ermöglichen. Dafür ist es auf der einen Seite erforderlich, eine angemessene Internetanbindung für alle Nutzer zur Verfügung zu stellen. Auf der anderen Seite ist aber auch die nutzergerechte Gestaltung der Benutzerschnittstellen zu beachten, z.B. die Zugänglichkeit und Bedienbarkeit für ältere Menschen oder Menschen mit Behinderung.

Eine ergonomische und funktionsgerechte Benutzerschnittstelle ist heute eine wichtige Grundvoraussetzung für die Akzeptanz eines Softwareproduktes. Dabei hängen die Anforderungen sehr stark von den entsprechenden Nutzergruppen und ihren Bedürfnissen ab. Eine Webseite einer öffentlichen Behörde muss anders gestaltet sein als das Benutzerinterface eines Administrators für die Serverkonfiguration. Trotz aller Unterschiedlichkeit der Benutzerschnittstellen sollte es, neben der Umsetzung der funktionalen Anforderungen, auch immer das Ziel sein, Nutzern die Bedienung der Software so einfach und angenehm wie möglich zu gestalten.

3.1. Benutzerschnittstellen unterscheiden und Zukunftstrends präsentieren

➢ **Sie sollen verschiedene Benutzerschnittstellen unterscheiden und zukünftige Entwicklungen präsentieren.**



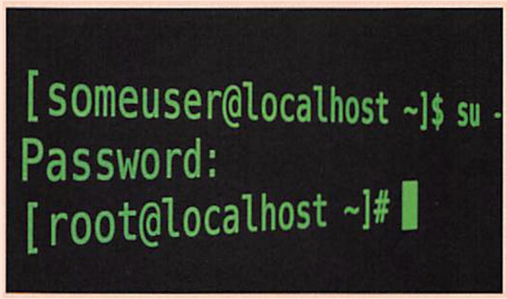
Bevor eine Benutzerschnittstelle designt und implementiert wird, sollte geklärt werden, um welche Art von Benutzerschnittstelle es sich handelt. Allgemein betrachtet gehören die Benutzerschnittstellen erst einmal zu den Mensch-Maschinen-Systemen (MMS), durch deren Hilfe der Mensch mit einer Maschine in Interaktion treten kann. Bezogen auf die Informatik ist es konkret die Mensch-Computer-Interaktion, welche hier betrachtet wird. Hier spricht man in diesem Zusammenhang auch von der Benutzeroberfläche oder dem User Interface (UI). Computergestützte Benutzerschnittstellen sind die Teile eines Computerprogramms, mit denen ein Benutzer interagieren kann.

Benutzerschnittstelle nach DIN EN ISO 9241-110
Eine Benutzerschnittstelle sind alle Bestandteile eines interaktiven Systems, welche Informationen und Steuerungselemente zur Verfügung stellen, mit deren Hilfe ein Benutzer eine bestimmte Arbeitsaufgabe mit dem interaktiven System durchführen kann.

Heutzutage verbinden die meisten Menschen eine computergestützte Benutzerschnittstelle mit einer grafischen Benutzeroberfläche, welche mit der Maus und der Tastatur oder auch mit der Hand per Touchscreen bedient wird. Diese Formen der Interaktion mit dem Computer sind auch am meisten verbreitet. Trotzdem gibt es noch andere Formen der Interaktion, welche, obwohl teilweise schon eingesetzt, sich noch in der Entwicklung befinden, aber in der Zukunft eine große Rolle spielen könnten. Hier reicht das Spektrum von der Spracheingabe bis hin zu Gehirn-Computer-Schnittstellen. Nachfolgend werden einige Arten von Benutzerschnittstellen näher betrachtet.

Arten von Benutzerschnittstellen

Befehlszeilenschnittstelle oder Kommandozeile (Command Line Interface, CLI)



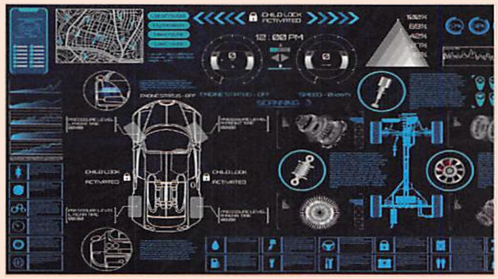
Bei dieser Schnittstelle interagiert der Benutzer mit dem Computer, indem er per Tastatur Befehle in ein Terminal eingibt und diese dann ausführt. Die Ergebnisse werden dann auch in diesem Terminal angezeigt. Zu Beginn waren CLIs die einzige Form der Benutzerschnittstelle. Aber auch heute wird diese Form der Interaktion noch viel verwendet, z. B. im Bereich der Systemadministration. Für nahezu alle Betriebssysteme gibt es eine solche Kommandozeilenebene (z. B. Windows: Power Shell, Linux: Shell). Dabei steht die Funktionalität im Vordergrund und nicht die optische Gestaltung. Auf diese Weise können die notwendigen Administrations- und Wartungsarbeiten effektiv ausgeführt werden. Im unmittelbaren Anwenderbereich ist das CLI aber nahezu vollständig verschwunden. Im Vergleich zu GUIs verbraucht diese Art von Schnittstelle wenig Ressourcen und ist daher auch für weniger leistungsstarke Computer geeignet.

Zeichenorientierte Benutzerschnittstelle (Text User Interface, TUI)



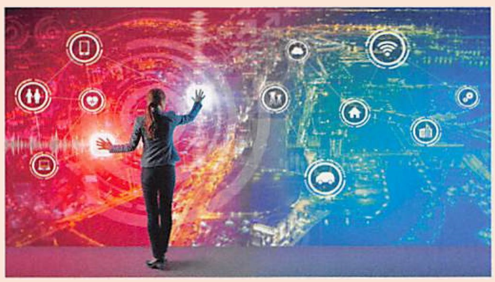
Diese Form der Benutzerschnittstelle stellt eine Erweiterung der CLI dar. Hier werden keine Befehlseingaben mehr vom Benutzer verlangt. Die Programme bieten textbasiert Menüs an, welche in der Regel mit der Tastatur bedient werden. Dadurch, dass keine Befehle mehr erlernt werden mussten, konnte die Bedienung der Programme viel schneller erlernt werden. Auch heutzutage haben textbasierte Programme noch eine treue Gemeinde, z. B. im Linux-Be reich. Im Allgemeinen aber ist diese Art von Benutzerschnittstelle vom Desktop des Endanwenders verschwunden

Grafische Benutzeroberfläche (Graphical User Interface, GUI)



Grafische Benutzeroberflächen sind heutzutage der Standard bei der Anwendersoftware. Hier lassen sich anspruchsvolle, optisch ansprechende und komplexe Oberflächen gestalten. Bedient wird eine GUI i.d.R. mit der Maus und der Tastatur. Je nach Anwendung kommen aber auch andere Eingabegeräte zum Einsatz, z.B. der Joystick bei Spielen. Interaktionselemente sind beispielsweise Icons, Buttons, Textfelder; die Programmfenster können in variabler Größe und Position geöffnet werden. Je nach Betriebssystem gibt es verschiedene Bibliotheken, welche zum Einsatz kommen. Als Beispiele seien hier das Net-Framework von Microsoft oder das Plasma Framework angeführt. Im Vergleich mit den CLIs verbrauchen die GUIs viel mehr Ressourcen und erfordern mehr Rechenpower.

Natürliche Benutzerschnittstellen (Natural User Interface, NUI)



Bei den natürlichen Benutzerschnittstellen handelt es sich i.d.R. auch um grafische Benutzeroberflächen. Allerdings erfolgt hierbei die Interaktion nicht mehr mithilfe einer Maus, sondern z. B. mithilfe eines Touchscreens, welcher empfindlich auf Finger- und Handbewegungen reagiert. Im Bereich der Industriesoftware oder bei Smart-Home-Systemen haben die Touch-Oberflächen die Interaktion mit Maus und Tastatur komplett ersetzt. Eine weitere Entwicklung ist die gestenbasierte Bedienung. Diese lehnt sich an natürliche, gewohnte Bewegungen an und ermöglicht so einen intuitiven Umgang mit interaktiven Geräten. In diesem

Zusammenhang finden VR-Brillen ihren Einsatz. Durch sie haben Benutzer den Eindruck, sich komplett in der virtuellen Welt zu befinden. Die Oberfläche wird bei einem NUI nicht mehr hierarchisch organisiert, sondern die gewünschten Inhalte werden auf den Nutzer zubewegt. Je nach Bedeutung werden die Bildschirmobjekte vergrößert, verkleinert oder entsprechend platziert. Hier können Benutzer die Inhalte nach ihren Bedürfnissen zusammenstellen.

Sprachbasierte Benutzerschnittstellen (Voice User Interfaces, VUI)



Bei den sprachbasierten Benutzerschnittstellen kommunizieren Benutzer mit der Software, indem sie Befehle mittels Sprache eingeben. Ausgaben erfolgen weiterhin über eine grafische Anzeige oder über eine synthetische Stimme, welche die Ausgaben vorliest. Noch ist die Interaktion über reine Spracheingabe und -ausgabe (z. B. Alexa, Siri, Cortana) nicht weit verbreitet. In der Regel handelt es sich um eine Kombination aus Sprach- und Texteingabe, d.h., es kommen auch hier noch die Maus und die Tastatur zum Einsatz. Rein sprachgesteuerte Benutzerschnittstellen sind immer dann sinnvoll, wenn Benutzer ihre Hände nicht umfänglich für die Interaktion mit dem System nutzen können (Arbeit) oder möchten (Smart Home).

Organische Benutzerschnittstelle (Organic User Interface, OUI)



Bei einer organischen Benutzerschnittstelle handelt es sich um eine nicht flache bzw. eine biegbare Anzeige. In den traditionellen GUIs sind Eingabe und Ausgabe physisch getrennt. Die Ausgabe wird grafisch auf dem Bildschirm auf der Grundlage der Eingabe eines Steuergeräts erzeugt. Bei den OUI ist die Eingabe gleich die Ausgabe, d.h., die Displayoberfläche und ihre physischen Verformungen sind immer der Ort der Benutzerinteraktion. Die Formen der Oberflächen werden durch ihre Funktionalität bestimmt und sollten so gewählt werden, dass die gewünschte Funktionalität optimal unterstützt wird, z.B. das Zusammenfalten eines dünnen Tablet-PC zu Transport- und Verstauungszwecken. Bei den OUI gibt es verschiedene Formen: flexible und verformbare Benutzeroberflächen, geformte Benutzeroberflächen (z. B. Kugeln, Zylinder, Kugelschreiber) und angesteuerte Benutzeroberflächen (z. B. 3-D-Voxel zum physikalischen Darstellen vollständiger 3-D-Bilder).

Gehirn-Computer-Schnittstelle (Brain Computer Interface, BCI)

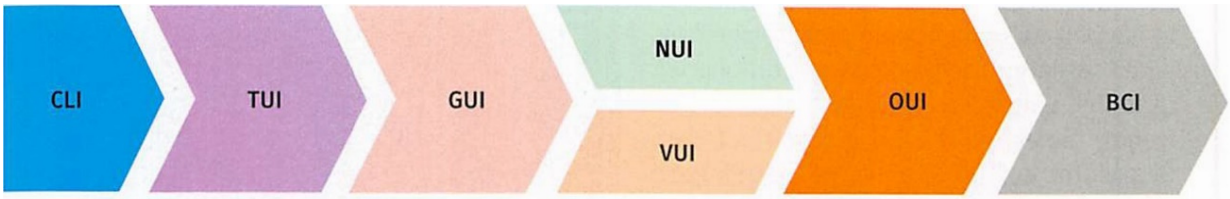


Bei dieser Form der Benutzerschnittstelle wird die Bedienung eines Computers über Gedanken ermöglicht. Allerdings steckt diese Entwicklung noch in den Kinderschuhen. Heutzutage wird diese Art der Benutzerschnittstelle vor allem für die Unterstützung von Menschen mit körperlicher Behinderung eingesetzt. Hier werden die Gedankenströme u.a. zur Steuerung von Prothesen, Rollstühlen oder als Sprechhilfe verwendet.

In den Anfängen des Computers bestanden die Benutzerschnittstellen überwiegend aus Lochkarten, Lochkartenlesern und Lochkartenstanzern. Eine wirkliche Benutzerschnittstelle kam dann mit der Befehlszeilenschnittstelle (CLI) auf, welche am Anfang nichts weiter als ein Bildschirm mit einer Eingabezeile und einem Eingabe-Prompt war. Hier wurden die Befehle in den Computer per Tastatur eingegeben.

Im Laufe der Zeit entwickelten sich textbasierte Benutzerschnittstellen, bei denen Menüs eine Reihe von Auswahlmöglichkeiten dem Benutzer für die Interaktion mit Computern zur Verfügung stellten. Das Konzept von GUIs im heutigen Sinne wurde schon in den 1970er-Jahren entwickelt. Allerdings setzte es sich erst langsam durch, weil die damaligen Computer noch zu langsam waren.

Einen größeren Kreis von Anwendern erreichte das GUI-Konzept erst durch die populäreren Computer von Apple in den 1980er-Jahren. Microsoft übernahm später dieses Konzept für sein Windows. Damit begann der Siegeslauf der GUIs. Die Einführung von Windows 95 war dabei eine weitere wichtige Etappe. Durch eine neue Startleiste und ein verändertes Fensterkonzept wurden die Bedürfnisse des Endkunden stärker als bisher berücksichtigt. Trotzdem wirken aus heutiger Sicht die vergangenen GUIs sehr unausgereift. Mit zunehmender Rechenleistung und dem Aufkommen von Multimedia wurden Ton, Sprache, bewegte Bilder und virtuelle Realität zunehmend in die grafischen Benutzeroberflächen integriert. Die Eingabemöglichkeiten um Spracheingaben, Gesten und Berührungen (Touchscreens) zu erweitern, war der nächste Entwicklungsschritt. Mit der Zeit entwickelten sich eigenständige Benutzeroberflächen, welche ihren Schwerpunkt auf diese Form der Interaktion legen, z. B. Smartphones.



Historische Entwicklung von Benutzeroberflächen

Durch die vielfältigen Möglichkeiten der Interaktion entwickelten sich bis heute auch eine Unmenge an Benutzerschnittstellen. Diese Vielfältigkeit ist auf der einen Seite ein Vorteil, auf der anderen Seite aber auch ein Nachteil, weil Benutzer sich immer wieder neu auf unbekannte Benutzerschnittstellen einstellen müssen. Dieses kann schnell überfordern, zumal wenn die Benutzerschnittstellen nicht intuitiv und leicht zu bedienen sind. Anwender möchten keine Software bedienen, sondern eine Aufgabe erledigen. Die Bedienung sollte sie dabei daher so optimal wie möglich unterstützen. Hier ist es für Entwickler wichtig, allgemeine Gestaltungsregeln und Kriterien für Benutzeroberflächen zu kennen, welche zu einfachen und bedienungsfreundlichen Benutzerschnittstellen führen und auch auf den unterschiedlichsten Plattformen angewandt werden können. Zudem sollten sie unabhängig von aktuellen „Modetrends" bei der Schnittstellengestaltung sein.

❖ Konkrete Kriterien für eine ergonomische Gestaltung von Software sind u.a. in der ISO 9241 -110 zu finden.

✓ Kompetenzcheck

1. Geben Sie an, welche Aussagen richtig sind.
   a. Über eine Benutzerschnittstelle kann ein Anwender mit dem Computer interagieren.
   b. Das CLI gehört zu den frühen Benutzerschnittstellen in der Computertechnik.
   c. Das CLI wird heutzutage nicht mehr genutzt.
   d. Zeichenbasierte Benutzerschnittstellen werden hauptsächlich mit der Maus angesprochen.
   e. Das Konzept eines GUIs wurde als erstes von Microsoft im Betriebssystem Windows umgesetzt.
   f. Die Spracheingabe und die Interaktion per Touchscreen erweitern die Interaktionsmöglichkeiten, u.a. von GUIs.
   g. Moderne Benutzerschnittstellen bestehen u.a. aus VR-Brillen und lassen sich auch durch Gesten steuern.
   h. Organische Schnittstellen sollen Interaktionsoberflächen ermöglichen, welche die unterschiedlichsten Formen annehmen können.
   i. BCI ermöglichen auch Menschen mit Behinderung die Interaktion mit Computern und Geräten.
2. Entwickeln Sie in Partnerarbeit Ideen, wie neben den klassischen Benutzerschnittstellen zukünftige Benutzerschnittstellen für die Software des Modellkunden „Yachthafen Resort" aussehen könnten. Beschreiben Sie dazu die Zielgruppe, die Form der Interaktion und welche Vorteile diese für die Zielgruppe bildet. Stellen Sie im Anschluss Ihre Ideen in der Klasse vor.
3. Bearbeiten Sie im Handout die Lernsituation 3 mit den Aufgaben 1 und 2.

3.2. Kriterien zur Gestaltung und Bewertung von Benutzerschnittstellen präsentieren

- **Sie sollen sich grundlegende Kenntnisse über Interaktionsprinzipien und das UX Design von Benutzerschnittstellen erarbeiten.**

Jeder Softwareentwickler und jeder Webdesigner sollte sich mit den grundlegenden Kriterien zur Gestaltung von Benutzerschnittstellen auskennen. Denn um eine Software erfolgreich auf dem Markt zu etablieren, muss die Benutzung des Softwareproduktes so intuitiv und einfach wie möglich gestaltet werden.



Eine Verordnung, in der grundlegende Gestaltungsregeln und Kriterien zur Bewertung von Benutzerschnittstellen aufgeführt werden, ist die ISO-Norm 9241-110, die 2020 in der zweiten Auflage unter dem Titel „Interaction Principles" veröffentlicht wurde. Im Kern hat sich diese Norm im Vergleich zur ersten Auflage nicht stark geändert. Sie enthält sieben sogenannte Interaktionsprinzipien (bisher: Dialogprinzipien), welche bei der Bewertung, dem Entwurf oder der Umsetzung jeglicher Benutzungsschnittstelle anwendbar sind. Dabei spielt es keine Rolle, ob es sich um Software, Hardware oder eine Kombination aus beidem handelt. Jedes Interaktionsprinzip besitzt wiederum einige Unterkategorien, welche die Inhalte genauer beschreiben.

Interaktionsprinzipien und deren Ausprägungen nach der ISO-Norm 9241-110

Interaktionsprinzip	Ausprägungen
Aufgabenangemessenheit	**Identifizierbarkeit der unterstützten Aufgaben** Das System hilft dem Benutzer festzustellen, inwieweit seine Aufgaben durch das System unterstützt werden.
	Aufwandsoptimierung bei der Aufgabenerledigung Das System lässt den Benutzer die Schritte machen, die aus Benutzersicht wirklich erforderlich sind, und erspart dem Benutzer jegliche überflüssigen Schritte.
	Standardauswahlmöglichkeiten (Defaults) Das System bietet Defaults, wo immer dies sinnvoll möglich ist.
Selbstbeschreibung	**Vorhandensein und Offensichtlichkeit von Informationen** Angezeigte Informationen sind für den Benutzer unmittelbar auffindbar und verständlich.
	Eindeutige Anzeige des Systemstatus Der Benutzer erkennt jederzeit, ob das System arbeitet und was es tut bzw. ob Eingaben oder Auswahlen durch den Benutzer erforderlich sind.
Steuerbarkeit	**Unterbrechbarkeit** Der Benutzer kann die Aufgabenerledigung unterbrechen, um an anderer Stelle weiterzuarbeiten bzw. die Aufgabenerledigung später fortzusetzen.
	Flexibilität Der Benutzer kann unterschiedliche Interaktionsmittel wählen, um zum Ziel zu kommen.
	Individualisierbarkeit Der Benutzer kann die Benutzungsschnittstelle so anpassen, dass eine optimale Aufgabenerledigung, gemäß individueller Erfordernisse möglich ist.

Interaktionsprinzip	Ausprägungen
Erwartungskonformität	**Systemverhalten/-reaktionen wie erwartet** Das System verhält sich immer so, wie die Benutzer es aus der aktuellen Aufgabe heraus erwarten.
	Konsistenz (intern und extern) Das System ist bezüglich seiner Darstellung und seines Verhaltens einheitlich gestaltet (interne Konsistenz) und nutzt, wo immer sinnvoll, auch Darstellungsformen und Verhaltensweisen, die dem Benutzer aus anderen Systemen vertraut sind (externe Konsistenz).
	Änderungen im Nutzungskontext werden erkannt Das System stellt fest, dass sich im Nutzungskontext (z.B. physische Umgebung) die Bedingungen verändert haben und passt sich hieran an.
Robustheit gegen Benutzungsfehler	**Benutzungsfehlervermeidung** Durch das System vorhersehbare Benutzungsfehler werden durch geeignete Gestaltung unterbunden.
	Benutzungsfehlertoleranz Gemachte Benutzungsfehler werden vom System erkannt und wenn möglich automatisch korrigiert bzw. Eingaben und Auswahlen trotz erkannter Fehler richtig verarbeitet.
	Fehlermanagement Bei gemachten Benutzungsfehlern, die systemseitig nicht korrigierbar sind, wird der Benutzer bei der effizienten Fehlerbehebung unterstützt.
Benutzerbindung	**Motivation** Bedienfunktionen sind ansprechend und einladend gestaltet.
	Vertrauenswürdigkeit Das System vermittelt dem Benutzer glaubwürdig, dass durch die Nutzung keine Risiken für den Benutzer entstehen.
	Integration der Benutzer Benutzer können direkt an der Benutzungsschnittstelle Verbesserungsvorschläge machen und werden informiert, wie mit dem jeweiligen Verbesserungsvorschlag umgegangen wurde.
Erlernbarkeit	**Unterstützung beim Entdecken von Bedienfunktionen** Das System ist so gestaltet, dass der Benutzer aktiv auf unbekannte Bedienfunktionen aufmerksam gemacht wird und so effizienter mit dem System arbeiten kann.
	Unterstützung beim Ausprobieren von Bedienfunktionen Das System ist so gestaltet, dass der Benutzer bisher unbekannte Bedienfunktionen ausprobieren kann, ohne dass dies negative Konsequenzen hat.
	Unterstützung beim Erinnern und Wiedererkennen von Bedienfunktionen Das System ist so gestaltet, dass der Benutzer bei Bedienfunktionen, die er vor längerer Zeit genutzt hat bzw. die er nur gelegentlich nutzt, sofort erinnert, wie diese zu nutzen sind, und keine erneute Einarbeitung notwendig ist.

Um die Kriterien umzusetzen und optimale Benutzerschnittstellen zur Verfügung zu stellen, fließen die Erkenntnisse von unterschiedlichen wissenschaftlichen Disziplinen wie Psychologie, Arbeitswissenschaft oder Design beim Entwurf und der Umsetzung der Benutzerschnittstellen mit ein. Für die Beschreibung der Qualität der Interaktion zwischen dem Menschen und dem Computer hat sich auch der Begriff „Usability" etabliert. Ein weiterer Begriff in diesem Zusammenhang, welcher auch in der ISO 9241-210 aufgeführt ist, ist die „User Experience". Die User Experience (UX) ist ein wesentliches Kriterium, welches bei der Akzeptanz eines Softwareproduktes durch den Kunden eine Rolle spielt. Ein Verfahren, welches die UX bei der Entwicklung einer Benutzerschnittstelle in den Mittelpunkt stellt, ist das User Experience Design (UX Design).

User Experience
Unter User Experience werden die Nutzererfahrung bzw. die Wahrnehmungen und Reaktionen einer Person verstanden, die aus der tatsächlichen und der erwarteten Benutzung eines Produkts, eines Systems oder einer Dienstleistung resultieren.



Das UX Design befasst sich mit der Analyse, Gestaltung und Optimierung der User Experience. Zum Beispiel wird versucht, komplexe Systeme einfach darzustellen, um die Nutzer so bequem und schnell wie möglich an das gewünschte Ziel zu bringen. Für ein gutes Softwaresystem ist heutzutage gutes UX Design unabdingbar. Leider gibt es viele Beispiele für ein missglücktes UX Design, z. B. unübersichtliche Webseiten oder komplizierte Fahrkartenautomaten. Eine gelungene User Experience ist hingegen nicht besonders auffällig. Je besser die UX ist, umso weniger wird sie wahrgenommen. Ein gutes UX Design bringt mehrere Vorteile mit sich. Zum einem wechseln z. B. zufriedene Kunden seltener zur Konkurrenz und bei Webauftritten wird der Umsatz gesteigert. Zum anderen wird auch der Kundendienst entlastet, weil Nutzer weniger Fragen zur Handhabung der Software haben.

Das UX Design ist nicht mit einem Ul Design zu verwechseln. Der Unterschied besteht darin, dass sich ein Ul-Designer darum kümmert, dass ein digitales Produkt ästhetisch aussieht. UX-Designer bemühen sich darum, dass ein Produkt gut zu benutzen ist. Das UX-Design ist ein multidisziplinäres Feld, das Erkenntnisse u.a. aus visuellem Design, der Mensch-Computer-Interaktion, der Psychologie und Softwareentwicklung miteinander verbindet. Die folgenden Komponenten sind grundlegende Bestandteile von UX Design.

Bestandteile des User Experience Designs (UX Design)	
Visuelles Design	Visuelles Design (Ul Design) beschreibt die Ästhetik des Softwareprodukts. Es stellt die abschließende Ebene des User-Experience-Design-Prozesses dar. Dazu werden im visuellen Design entsprechend Farben, Bilder und Symbole eingesetzt. Hier gilt: Weniger ist meist mehr. Das Design sollte die Funktionalität der grafischen Benutzeroberfläche unterstützen.
Informationsarchitektur	Die Informationsarchitektur umfasst das Strukturieren von Informationen, sodass der Nutzer sie möglichst intuitiv finden kann. Um Inhalte den Nutzern zugänglich zu machen, erstellen Informationsarchitekten entsprechende Navigationskonzepte.
Interaktionsdesign	Das Interaktionsdesign ist der wichtigste Teil des UX Design, welches das Ziel hat, Softwareprodukte zu entwickeln, die intuitiv und effizient nutzbar sind. Dazu befassen sich Interaktionsdesigner ausführlich mit der Nutzerforschung, um Erwartungen und Bedürfnisse ihrer Zielgruppen zu erfassen.
Usability (Gebrauchstauglichkeit)	Die Usability beschreibt, inwiefern ein Softwareprodukt Nutzer dabei unterstützt, ihre Ziele in einem bestimmten Nutzungskontext zu erreichen. Je effektiver und einfacher die Bedienung ist, desto höher ist die Usability. Die Usability kann durch Usability-Tests objektiv gemessen werden. Aus diesen Tests können aber nur bedingt brauchbare Aussagen abgeleitet werden, da die Usability die subjektive Wahrnehmung von Nutzern nicht berücksichtigt.
Zugänglichkeit	Die Zugänglichkeit eines Softwaresystems beschreibt, wie leicht dessen Nutzung fällt und wie dessen Erreichbarkeit ist. Sie trägt dazu bei, die mit einem System verbundene Lernkurve zu verkürzen. Zugänglichkeit kann in vielen Kontexten auch mit der Einfachheit einer Anwendung für Menschen mit Behinderung in Verbindung gebracht werden (siehe Barrierefreiheit, S. 45).

Um eine positive User Experience zu erreichen, gibt es eine Reihe von Designprinzipien. Die nachfolgenden Prinzipien stellen eine Auswahl aus dieser Reihe dar. Sie bieten Entwicklern die Möglichkeit eines einfachen Einstiegs in die Fragen des Designs von Benutzerschnittstellen und können bei Bedarf jederzeit erweitert und verfeinert werden.

Grundprinzipien des UX Designs

Vermeidung von Informationsflut
Es sollte vermieden werden, dass der Benutzer sich zu viele Informationen merken muss, um z. B. nachfolgende Aktionen auszuführen. In der Regel kann sich der Mensch zwischen fünf bis sieben verschiedene Informationen sicher merken.

Konsistenz
In ähnlichen Situationen sollten die Aktionssequenzen immer gleich gestaltet sein (z. B. Position, Layout, Farbe, Schriftart von Elementen).

Universelle Bedienbarkeit
Die Software sollte durch jeden Benutzer schnell und einfach bedient werden können, egal welche Voraussetzungen er mitbringt. Die Benutzer können sich beispielsweise in Alter oder Erfahrung unterscheiden.

Informative Rückmeldungen
Die Software sollte ausreichend Feedback über aktive Aktionen oder den Systemstatus liefern.

Abgeschlossene Dialoge
Dialoge sollten so gestaltet sein, dass Benutzern jederzeit bewusst ist, wann eine Aktion, Funktion oder Befehlskette gestartet wird und wann diese abgeschlossen ist.

Einfache Fehlerbehandlung
Bei Programmfehlern sollten Nutzer verständliche Informationen über potenzielle Ursachen bekommen. Sehr wichtig ist es dabei, einen Ausweg anzubieten, um in den normalen Programmbetrieb zurückzuwechseln.

Widerrufbarkeit von Aktionen
Nutzern soll die Möglichkeit gegeben werden, getätigte Aktionen wieder rückgängig zu machen.

Kontrollvermittlung
Dabei sollte das Programm so gestaltet sein, dass Nutzer agieren und das Programm reagiert und nicht umgekehrt.

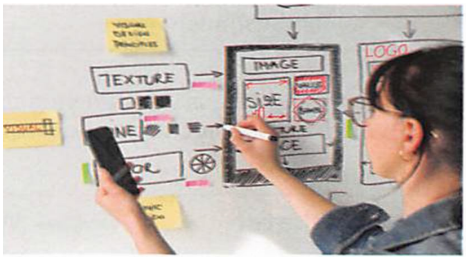
✓ Kompetenzcheck

1. Geben Sie an, welche Aussagen richtig sind.
   a. Selbstbeschreibung und Aufgabenangemessenheit gehören zu den Interaktionsprinzipien.
   b. Wenn der Speichern-Button für Daten im Programm immer an der gleichen Stelle liegt, dann ist das Programm erwartungskonform.
   c. Wenn die Schaltfläche zum Minimieren eines Fensters eines Windows-Programms links oben liegt, dann ist das Programm erwartungskonform.
   d. Durch durchdachte Bedienungsmöglichkeiten wird u.a. eine hohe Motivation erzeugt, das Programm zu nutzen. Dies führt wiederum zu einer höheren Benutzerbindung.
   e. Beim UX-Design wird sich vornehmlich auf die ästhetische Gestaltung der Benutzerschnittstelle konzentriert.
   f. Das visuelle Design ist ein Bestandteil des UX-Designs von Softwaresystemen.
   g. Die Zugänglichkeit beschreibt, wie leicht ein Softwaresystem zu benutzen ist und wie gut dessen Erreichbarkeit ist.
2. Vertiefen Sie Ihr Wissen über Interaktionsprinzipien und UX-Design, indem Sie die Aufgaben 3 bis 5des Kapitels 3 im Handout bearbeiten.

3.3. Den Designprozess und Designwerkzeuge präsentieren

➢ **Sie sollen sich den Prozess des Designens von Benutzerschnittstellen erarbeiten und präsentieren.**

Durch die wachsende Bedeutung der Benutzerschnittstelle für die Software ist der Prozess der Entwicklung von dieser zu einem eigenständigen Teilprozess in dem gesamten Entwicklungsprozess der Software geworden. Natürlich bilden auch hier die funktionalen Anforderungen an die zukünftige Benutzerschnittstelle die Grundlage. Allerdings sollte der Designprozess auch so gestaltet sein, dass eine möglichst gute User Experience erzielt werden kann. Als Grundlage dafür eignet sich ein iterativer Ansatz, wie er aus dem Design Thinking bekannt ist.

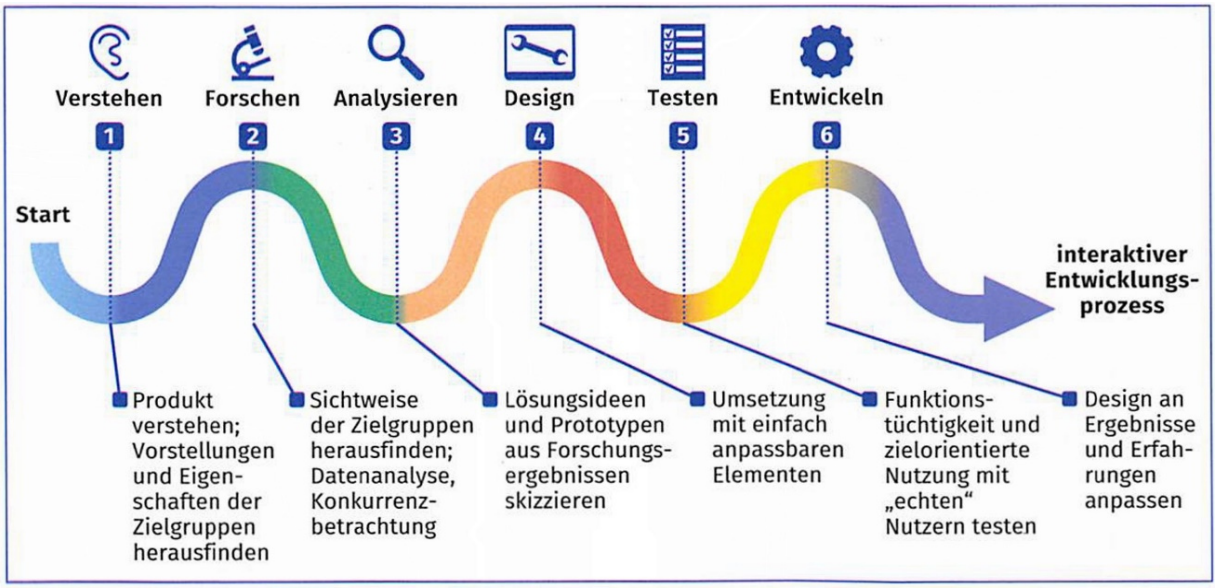




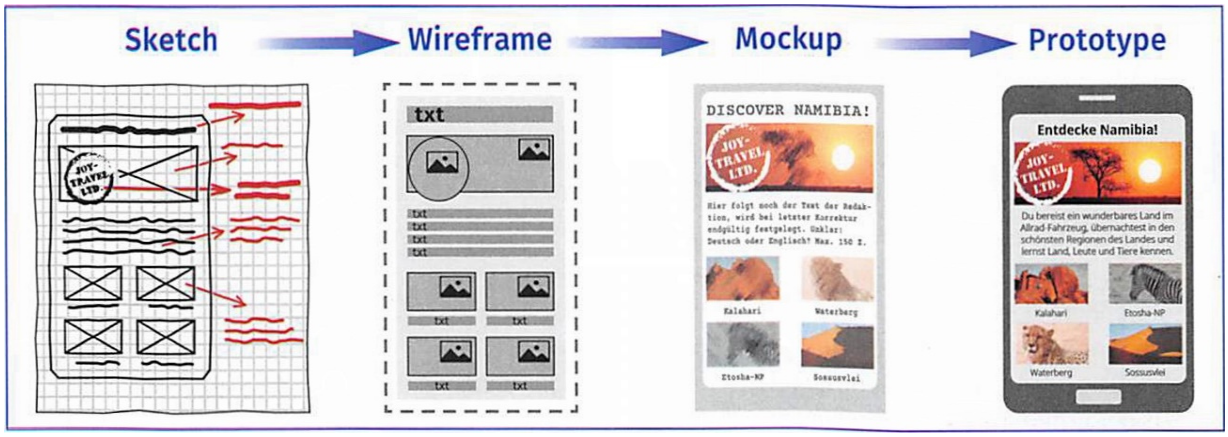
Design Thinking

Design Thinking ist eine Vorgehensweise, bei der die Wünsche und Bedürfnisse des zukünftigen Nutzers eines Systems im Mittelpunkt des Entwicklungsprozesses stehen. Es wird versucht, das Problem aus der Sicht der Nutzer zu betrachten und entsprechende Lösungen zu finden. Abgeleitet ist das Design Thinking aus der Arbeitsweise von Designern, welche einem intuitiven Prozess folgen, der im Kern auf Beobachtung und einer hohen Nutzerzentrierung basiert. Design Thinking eignet sich für die unterschiedlichsten Anwendungsgebiete. Auch für den Bereich der Digitalisierung ist Design Thinking eine geeignete Vorgehensweise, um digitale Produkte und Geschäftsmodelle zu entwickeln.

Aus diesem Ansatz wird der Prozess des UX Design abgeleitet, welcher sich in drei große Bereiche einteilen lässt. Der erste Bereich besteht aus dem Verstehen, dem Forschen und dem Analysieren. Hier werden alle funktionalen und nicht funktionalen Anforderungen, die Zielgruppe und deren Bedürfnisse und eventuell vorhandene Konkurrenzprodukte untersucht und ausgewertet. Aus diesen Erkenntnissen werden dann erste Ideen und Entwürfe entwickelt. Der zweite große Bereich in diesem Prozess ist das Designen der Schnittstelle und die Umsetzung. Dabei werden die Erkenntnisse aus den vorhergehenden Schritten berücksichtigt. Nach der Implementierung erfolgt das Testen der Benutzerschnittstelle. Dieses kann sowohl intern als auch extern in Form von Betatests (siehe Jahrgangsband 1, Modul 5) geschehen. Getestet werden sollten alle Kriterien, welche für die zukünftige Benutzerschnittstelle wichtig sind. Von besonderem Interesse dabei ist die Rückmeldung der Benutzer über ihre User Experience. Damit ist der Prozess aber nicht abgeschlossen, sondern er wird wiederholt. Dabei wird im nächsten Durchlauf das Design auf Grundlagen der Ergebnisse der Testphase angepasst.



In dem Designprozess werden verschiedene Formen von Entwürfen angefertigt. Zu nennen sind hier Sketches, Wireframes, Mockups und Prototype (siehe auch Jahrgangsband 2, Modul 8).



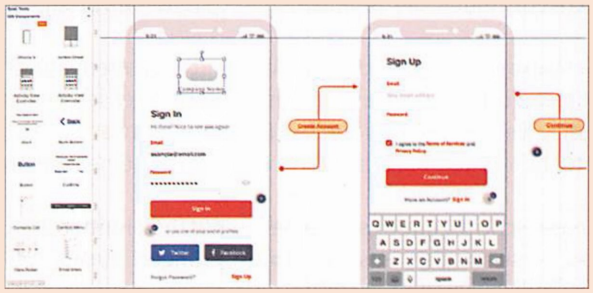
Für das Erstellen von Sketches, Wireframes und Mockups gibt es sehr viele Tools, welche die Entwickler bei der Arbeit unterstützen. Viele dieser Anbieter bieten neben einem kommerziellen Account auch einen kostenlosen, eingeschränkten Zugang an.

Beispiele für Tools für die Entwicklung von Wireframes und Mockups

Adobe XD
Adobe XD ist eine leistungsstarke und benutzerfreundliche vektorbasierte Design-Plattform u.a. für die Erstellung von Wireframes und Prototypen. Adobe XD ist auf Mac- und Windows- Systemen verfügbar.
www.adobe.com/de/products/xd.html

Mockplus
Mockplus ermöglicht es, alle Aspekte einer GUI zu entwerfen und basiert auf dem WYSIWYG- Modell. (What you see is what you get). Es bietet eine Reihe von spezifischen Vorlagen für verschiedene Geräte, wie Smartphones, Tablets oder PCs, welche per Drag & Drop an die eigenen Bedürfnisse angepasst werden können.
www.mockplus.com/ui-mockup-tool

Moqups
Moqups gehört zu den bekanntesten Mockup-Tools und ist rein webbasiert. Eine Offline-Nutzung ist nicht möglich. Es ist für iOS, Android, Web, MacOs und Windows geeignet. Hervorzuheben ist der tolle Support bei Moqups.
www.moqups.com/de/



FluidUI
FluidUI ist ein weiteres Wireframe Tool, welches allerdings auch zur Erstellung von Prototypen genutzt werden kann. Auch FluidUI kann nur online benutzt werden, wobei nicht alle Browser mit FluidUI kompatibel sind. Die Bedienung erfolgt auch hier einfach per Drag & Drop. Außerdem kann der Nutzer seine Ideen per Live-Video präsentieren und somit Kunden oder Mitarbeitern kommunizieren.
www.fluidui.com

Marvel App
Marvelapp ermöglicht es, Wireframes, Mockups und realistische Prototypen zu erstellen und steht online kostenlos mit etwas eingeschränkter Funktionalität zur Verfügung. Auch hier können die Ergebnisse mit der Kamera aufgenommen und präsentiert werden.
www.marvelapp.com/

✓ Kompetenzcheck

1. Geben Sie an, welche Aussagen richtig sind.
   a. Beim Design Thinking stehen die Wünsche und Bedürfnisse des Benutzers im Mittelpunkt der Designüberlegungen.
   b. Der Prozess des UX Design sollte maximal zweimal durchlaufen werden.
   c. Ein Sketch enthält einen ersten Entwurf und steht am Anfang des Designprozesses einer Benutzerschnittstelle.
   d. Wireframes enthalten keinen Text.
   e. Mockups können schon Bilder und Blindtexte enthalten.
2. Bearbeiten Sie im Handout Kapitel 3 die Aufgaben 6 und 7.

3.4. Grafische Benutzerelemente einer GUI unterscheiden und allgemeine Gestaltungskriterien präsentieren

- **Sie sollen sich grundlegendes Wissen über die verschiedenen Oberflächenelemente einer GUI und allgemeine Gestaltungskriterien erarbeiten und dieses Wissen bei konkreten Aufgabenstellungen anwenden.**



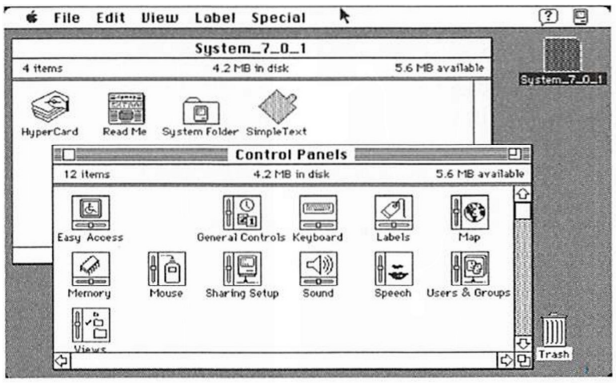
Um ein gutes UX Design zu gewährleisten, welches sich an den Interaktionskriterien für Benutzeroberflächen (siehe auch Kapitel 1.3.2) orientiert, sind alle Details einer Oberfläche wichtig. Diese umfassen das Layout und Farbschema, die Menüführung und die Gestaltung der einzelnen GUI-Elemente. Um hier eine gewisse Einheitlichkeit zu erreichen, sind verbindliche Vorgaben notwendig. Diese Vorgaben schließen beispielsweise optische Gestaltungsregeln für Oberflächenelemente, aber auch Gestaltungsregeln, welche die Barrierefreiheit der Benutzerschnittstelle betreffen, ein.

Der Schwerpunkt liegt in diesem Abschnitt vor allem auf der grafischen Gestaltung einer GUI-Oberfläche, weil diese Form derzeit am weitesten verbreitet ist. Dabei richten sich die Inhalte nicht an angehende Designer, welche professionelle Webseiten entwerfen. Es soll hier vielmehr neben der Entwicklung des eigentlichen Programms auch um den Entwurf einer ansprechenden GUI gehen.

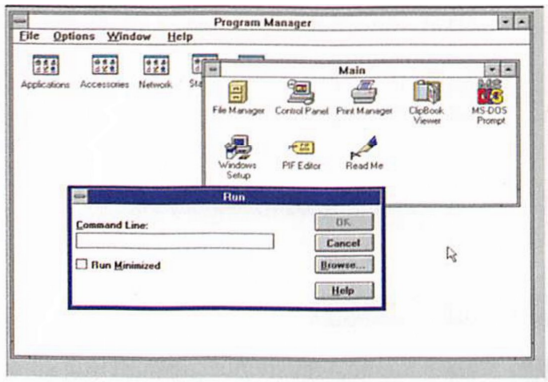
Ein erster Schritt zum Design einer Benutzerschnittstelle ist die Identifikation aller benötigten GUI-Elemente. Diese werden aus den Anforderungen abgeleitet und im Rahmen des Entwurfs der Benutzerschnittstelle zusammengestellt.

3.4.1. Grundlegende GUI-Elemente

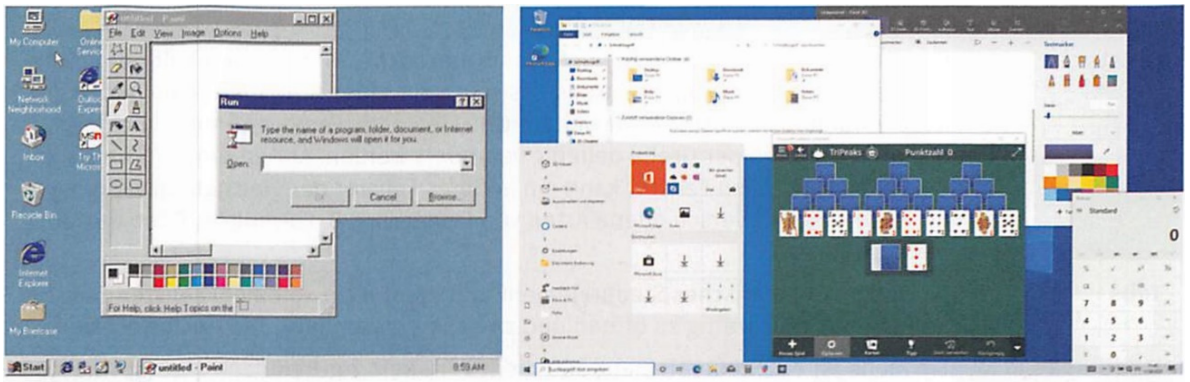
Die Geschichte der GUI begann bereits in der 1970er-Jahren. Hier wurden die ersten Konzepte entwickelt, welche bis heute immer mehr verfeinert wurden. Die grundlegenden grafischen Elemente waren schon damals vorhanden, wenn auch aus heutiger Sicht optisch wenig ansprechend.



Apple Macintosh 1984

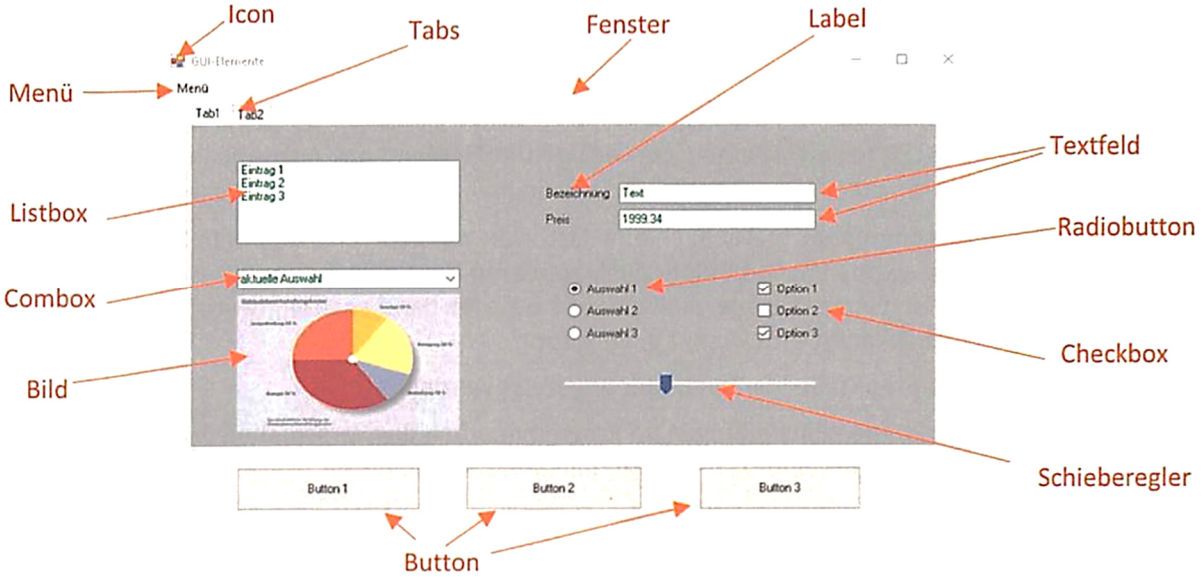


Windows 3.11





Wie die Beispiele der Vergangenheit zeigen, wird eine GUI in der Regel aus verschiedenen GUI-Elementen zusammengesetzt. Über die Zeit gesehen ist das Design zwar immer unterschiedlich, die Grundfunktionalität aber ist über die Jahrzehnte immer dieselbe geblieben. Die wichtigsten Elemente werden nachfolgend etwas näher vorgestellt.



Grundlegende GUI-Elemente	
Fenster	Generell ist ein Fenster ein Bereich einer Oberfläche, in dem ein Inhalt der Anwendung angezeigt wird, und stellt das Basiselement einer GUI-Oberfläche dar. In der Regel enthält ein Fenster dazu weitere GUI-Elemente. Fenster können minimiert, maximiert oder in der Größe beliebig verändert werden. Außerdem sind sie beliebig verschiebbar. Ein Fenster kann ein anderes Fenster der gleichen Anwendung enthalten. Es gibt verschiedene Arten von Fenstern, z. B. Formulare, Dialogboxen, Messageboxes.
Tabs	Ein Tab ist ein grafisches Steuerelement und dient dazu, in einem Fenster mehrere Dokumente gleichzeitig zu öffnen und zwischen diesen hin- und herzuwechseln.
Menü	Ein Menü ist ein Steuerelement und dient vor allem zur Benutzerführung. Im Menü kann der Benutzer aus einer Reihe von Optionen (Menüeinträgen) auswählen, welche anschließend ausgeführt werden. Jeder Menüeintrag kann wiederum andere Untermenüeinträge enthalten. Menüs können je nach Bedarf vollständig angezeigt oder teilweise ausgeblendet werden.
Icon	Ein Icon ist ein kleines Bild oder Symbol. Icons werden vielfältig verwendet und dienen meistens grafischen Untermalungen von Inhalten. Dabei können in vielen Fällen durch das Klicken auf das Icon Funktionalitäten des Programms aufgerufen werden.
Cursor	Ein Cursor ist eine grafische Darstellung eines Zeigers in einem GUI-System. Dieser wird in Zusammenhang mit einer Maus oder anderen Eingabegeräten verwendet und zeigt den aktuellen Ort auf der Oberfläche an, an dem die Eingaben dieser Geräte verarbeitet werden.
Textbox	Eine Textbox ist ein GUI-Element, in dem Benutzer Text zur weiteren Verarbeitung eingeben können oder in dem Text von dem Programm angezeigt wird.
Label	Ein Label ist ein GUI-Element, welches es ermöglicht, einen statischen, nicht editierbaren Text anzuzeigen.
Button	Der Button ist eine Schaltfläche, durch deren Betätigung Aktionen im Programm ausgelöst werden können.
Radiobutton	Radiobuttons stellen Benutzern verschiedene Auswahlmöglichkeiten zur Verfügung. Von den angebotenen Optionen kann nur eine ausgewählt werden.
Checkbox	Checkboxen stellen Benutzern verschiedene Auswahlmöglichkeiten zur Verfügung. Von den angebotenen Optionen können mehrere ausgewählt werden.
Listbox	Eine Listbox bietet eine Liste verfügbarer Elemente zur Auswahl an. Diese werden in Form einer Liste aufgeführt. Es kann mehr als ein Element ausgewählt werden.
Combobox	Eine Combobox bietet auch eine Liste von verfügbaren Elementen zur Auswahl an. Diese werden in Form einer ausklappbaren Liste aufgeführt. Es kann in der Regel nur ein Element ausgewählt werden, welches dann angezeigt wird. Der Rest der Möglichkeiten wird wieder eingeklappt.
Bild	An dieser Stelle können Bilder angezeigt werden.
Schieberegler	Hier können Einstellungen im Programm durch das Verschieben eines grafischen Elements auf einer Achse vorgenommen werden.

Mithilfe dieser GUI-Elemente können die unterschiedlichsten grafischen Benutzeroberflächen entwickelt werden. Die Verwendung dieser Elemente führt aber nicht automatisch zu einem guten Design. Außerdem stellt sich die Frage, wie einzelne Elemente grafisch zu gestalten sind.

3.4.2. Allgemeine Gestaltungsregeln

In vielen Fällen wird die Gestaltung der GUI-Elemente durch das verwendete Framework vorgegeben und kann so in die eigene Software übernommen werden. Hier spielt die Erwartungskonformität eine große Rolle. In anderen Fällen wird bewusst von den Vorgaben eines Frameworks abgewichen, um für die Software eine eigene

Identität zu schaffen. Als Beispiele können hier Computerspiele aufgeführt werden. Aber auch die Gestaltung von Webseiten erfolgt in der Regel relativ unabhängig von Vorgaben. Die Gestaltungsmöglichkeiten sind in diesem Bereich sehr vielfältig und es werden nicht selten professionelle Designer nur für den Entwurf solcher Oberflächen engagiert, welche dann bei Bedarf u. a. den „goldenen Schnitt" und andere Regeln der Proportionslehre oder des Designs in ihre Überlegungen einfließen lassen.

Im Folgenden finden sich allgemeine Gestaltungsregeln, deren Beachtung nicht viel Aufwand erfordert, aber mit deren Hilfe sich trotzdem gute GUI-Oberflächen entwickeln lassen.

Allgemeine Regeln für ein gutes UI Design von GUI-Oberflächen

Text
- Maximal zwei bis drei verschiedene Schriftarten für Überschriften, Text und interaktive Elemente verwenden.
- Unterstreichungen nur für Links verwenden.

Farben
- Licht kommt von oben. Dieses entspricht der Erfahrungswelt des Menschen. Dementsprechend sind GUI-Elemente oben hell und unten dunkler. Auch eventuelle Schatten sind so zu gestalten.
- Die Anzahl der verwendeten Farben einschränken.

Texteingabe
- Markierungen verwenden, um den Benutzer zur Eingabe der erforderlichen Daten zu führen.
- Benutzer können Daten in verschiedenen Formaten eingeben.

Steuerelemente
- Schaltflächen hervorheben, sodass Benutzer erkennen können, dass das Steuerelement anzuklicken ist.
- Gerade aktives Steuerelement von anderen Steuerelementen farblich oder formlich abheben.
- Deaktivierte Steuerelemente durchsichtig gestalten, nicht grau.
- Nicht relevante Steuerelement ausblenden oder zuklappen und erst bei Bedarf dem Benutzer zur Verfügung stellen.

Abstände
- Eine gute Bedienoberfläche benötigt viel Raum.
- Oberfläche nicht mit GUI-Elementen überladen.

Icons
- Icons sollten einfach gehalten werden, in Form und Farbe.
- Icons sollten eine klare Bedeutung für den Benutzer haben.
- Icons sollten alle den gleichen Stil besitzen.

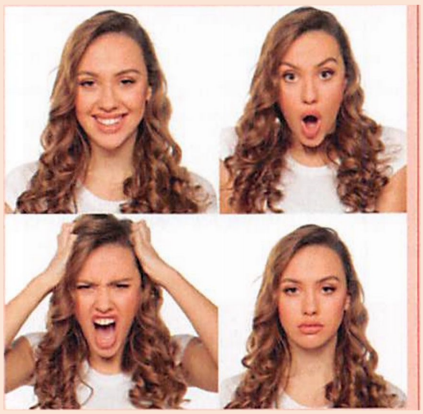
Navigation
- Durch die Anwendung sollte auch per Tastatur navigiert werden können.
- Die Anwendung sollte durch Shortcuts bedienbar sein.

3.4.3. Weitere Gestaltungsaspekte

Neben einem guten UI Design können für die Benutzeroberflächen auch noch weitere Aspekte eine entscheidende Rolle spielen, um bei Benutzern eine positive Assoziation mit dem Programm hervorzurufen. Ob die nachfolgenden Aspekte eine Rolle spielen und wenn ja, in welchem Umfang, hängt zum einen von den Anforderungen an die Software und zum anderen von der Zielgruppe ab, welche angesprochen werden soll.

Weitere Gestaltungsaspekte von Benutzeroberflächen

Emotionale Gestaltung



Bei der emotionalen Gestaltung einer Benutzeroberfläche geht es darum, Benutzer auch emotional anzusprechen und nicht nur rational. Softwarelösungen mit schlechten und frustrierenden Benutzerschnittstellen werden schnell abgelehnt. Deswegen ist bei der Gestaltung der Benutzeroberfläche u.a. darauf zu achten, dass diese einen positiven emotionalen Gesamteindruck hinterlässt. Dieser Eindruck hat mehr Einfluss auf die Bewertung des Gesamtproduktes als gemeinhin angenommen wird. Hier können motivierende, lernende, kreative und soziale Aspekte eine große Rolle spielen. Auch Gestaltungsgrößen wie Schriftart, Farbigkeit, Form, Größe, Helligkeit etc. spielen dabei eine Rolle.

Alle diese Parameter können bezogen auf die Zielgruppe aber sehr unterschiedlich ausfallen. Um die Zielgruppe genau anzusprechen, sind umfangreiche Analysen erforderlich. In vielen Fällen reicht es meistens schon aus, Benutzer nicht emotional zu verschrecken, wenn der Rest der Softwarelösung stimmig ist.

Klangliche Gestaltung

Auch die klangliche Gestaltung einer Softwarelösung kann in gewissen Bereichen eine Rolle spielen. Neben Anwendungen, bei denen die klangliche Gestaltung der Benutzerschnittstelle eine entscheidende Rolle spielt, z. B. bei Computerspielen, kann auch eine gelungene dezente klangliche Gestaltung bei anderen Softwareprodukten einen positiven Gesamteindruck auslösen. Dies beginnt schon bei der Wahl akustischer Signale, um eine Systemaktivität anzuzeigen. In vielen Fällen sind diese Signale vom Betriebssystem vorgegeben, können aber an gewissen Stellen in der Software gezielt ergänzt werden. Hierzu gehört auch die Gestaltung der Sprachausgabe. Eine angenehme Stimme eines Fahrassistenten ist hier z. B. von entscheidender Bedeutung.

Soziale Gestaltung



Ein weiterer Gestaltungsaspekt ist der Aspekt der sozialen Gestaltung der Benutzerschnittstelle. Dieser Aspekt gewinnt immer mehr an Bedeutung, weil die Vernetzung der digitalen Geräte immer weiter zunimmt. Es reicht nicht, dass man schöne Fotos mit seinem Smartphone machen kann. Viele Nutzer wollen diese auch schnell und unkompliziert mit anderen teilen können. In Spielen oder Chatrooms muss eine unkomplizierte und gesicherte Kommunikation mit anderen möglich sein. In vielen Softwareprodukten steht die soziale Interaktion im Mittelpunkt der Anwendung. Hier geht es vor allem um unkomplizierte und leicht zu bedienende Lösungen. Aber auch bei Softwareanwendungen, bei denen die soziale Interaktion nicht im Vordergrund steht, sollte der soziale Aspekt bei der Entwicklung der Benutzerschnittstelle zumindest angesprochen und diskutiert werden.

Corporate Design

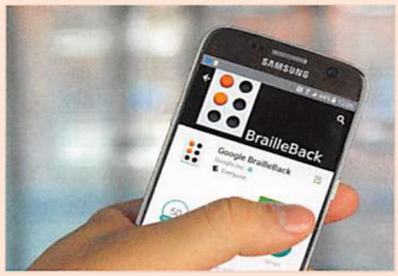


Corporate Design ist ein Teilbereich der Corporate Identity. Unter Corporate Identity werden die gebündelten Merkmale und Charakteristika eines Unternehmens zusammengefasst, durch die es definiert wird. Durch Corporate Identity ist ein Unternehmen von anderen Unternehmen unterscheidbar und es soll nach innen und nach außen als eine Einheit dargestellt werden. Bei der Entwicklung von Benutzerschnittstellen ist das Corporate Design der entscheidende Aspekt.

Einige dieser Maßnahmen zur Berücksichtigung des Corporate Design sind:

- Gestaltung eines Logos: Das Logo spielt eine zentrale Rolle im Corporate Design und wird oft als das Merkmal für CD angesehen. Das Logo sollte einzigartig, simpel und wiedererkennbar sein.
- Gestaltung des Schriftzugs: Um sich von Konkurrenten abzusetzen, sollte dafür am besten eine individuell angefertigte Schriftart verwendet werden. Der Schriftzug kann im Logo eingebunden sein, muss es aber nicht.
- Auswahl geeigneter Farben: Ein passendes Farbkonzept, welches das Unternehmen widerspiegelt, ist im Corporate Design nicht wegzudenken. In der Regel werden für das Farbkonzept zwei bis fünf verschiedene Farben verwendet.
- Auswahl geeigneter Gestaltungsraster: Ein Gestaltungsraster ist eine grobe Richtlinie zur Anordnung der verschiedenen Elemente. Es wird beispielsweise festgelegt, wo das Logo positioniert wird, wie groß einzelne Abbildungen sein können und welche Schriftgröße der Text haben soll.
- Auswahl geeigneter Bilder: Die Bilder sollten aussagekräftig sein und beim Kunden positive Assoziationen zum Produkt oder zum Unternehmen wecken.

Barrierefreiheit



Eine barrierefreie Gestaltung ist mehr als nur die Entwicklung von Benutzerschnittstellen für Menschen mit Körperbehinderung. Barrieren gibt es überall. Selbst Fremdsprachigkeit kann eine Barriere darstellen oder fortgeschrittenes Alter. Barrierefreiheit sollte dementsprechend grundsätzlich bei der Entwicklung der Benutzerschnittstellen mitberücksichtigt werden. Der Aufwand dafür ist geringer als viele glauben. Alle wichtigen modernen Betriebssysteme bieten dazu globale Funktionen an, welche Entwickler nutzen können. Dazu zählen beispielsweise die Bildschirmtastatur, die Unterstützung für Braille-Displays, kontrastreiche Darstellungen, Bildschirmlupen, das Vorlesen von Text oder die Spracheingabe von Befehlen. Weitere Möglichkeiten wären beispielsweise individuell anpassbare Schriftgrößen und Farben oder die gebärdensprachliche Interaktion mit dem Benutzer.

Mehrsprachigkeit



Manchmal ist es nötig, dass Programme in zwei oder mehr Sprachen bedient werden können. Dieses ist beispielsweise dann der Fall, wenn Unternehmen länderübergreifend tätig sind oder Mitarbeiter unterschiedlicher Nationalitäten beschäftigt werden. Um die Mehrsprachigkeit zu realisieren, ist es zum einen möglich, unterschiedliche Versionen der Software herauszubringen, welche jeweils eine andere Sprache berücksichtigen.

Zum anderen können alle benötigten Sprachen in die Software integriert und dann bei Bedarf entsprechend eingestellt werden. Dabei sind sämtliche Textanzeigen wie Beschriftungen und Hilfetexte in der entsprechenden Sprache umzusetzen. Manchmal ist es sogar notwendig, Währungseinheiten und Einheiten für Maße und Gewichte anzupassen.

3.4.4. Bedienung von grafischen Oberflächen mit anderen Interaktionsformen

Klassisch wird mit grafischen Benutzeroberflächen überwiegend mit der Maus und der Tastatur interagiert. In den letzten Jahren sind aber neue Interaktionsformen hinzugekommen, welche entweder in Kombination mit Maus und Tastursteuerung oder komplett eingeständig eingesetzt werden. Zu nennen sind hier die Touch-, Gesten- und Sprachsteuerung. Während die touchbasierte Steuerung vor allem im Smartphone- und Tablettbereich zum Standard geworden ist, befinden sich die Sprach- und Gestensteuerung noch in der Entwicklung.

Umsetzung von verschiedenen Interaktionsformen

Touchsteuerung
Die Touchsteuerung erlaubt die Bedienung der Oberfläche durch das Berühren der entsprechenden Elemente auf dem Bildschirm mit dem Finger. Dieses ersetzt quasi die Maussteuerung und der Finder wird zum „Cursor". Technisch ist hierbei keine Änderung zur Mausbedienung erforderlich. Ein Touch-Ereignis reagiert in dieser Art und Weise nahezu identisch wie ein Klick mit der Maus. Eventuell müssen die Steuerungselemente (Buttons, Listenauswahl usw.) in der Größe angepasst werden, da die Genauigkeit der „Fingerbedienung" geringer ist als die eines Mauszeigers. Deswegen sollte schon bei der Anforderungsspezifikation mit festgelegt werden, ob eine Touchbedienung vorgesehen ist, damit dieses beim Design der Oberfläche mitberücksichtigt werden kann.

Gestensteuerung
Heutzutage bieten entsprechende Frameworks Möglichkeiten für die Gestensteuerung von grafischen Oberflächen an, d. h., sie stellen entsprechende Events zur Verfügung. Allerdings müssen dafür die „normalen" GUI umfassend überarbeitet werden. Beispielsweise müssen Menüs durch interaktive und situativ eingeblendete Inhalte ersetzt werden. Es müssen Fragen nach der Form und der Größe von Steuerelementen geklärt werden und welche Gesten welche Interaktionen auslösen.

Sprachsteuerung
Die Sprachsteuerung hat in den letzten Jahren immer mehr an Bedeutung gewonnen. Möglich gemacht haben das verschiedene KI-Services, welche im Internet angeboten werden. Dadurch kann praktisch jede Software mit einer rudimentären Sprachsteuerung ausgestattet werden. Allerdings ist in vielen Fällen dafür eine Internetanbindung notwendig. Zunächst muss ein geeigneter Cloud- Service ausgewählt und in die eigene Software integriert werden. Danach ist die KI zu konfigurieren und die einfachen Sprachbefehle sind zu definieren. Für die Interaktion aus der eigenen Software heraus stehen dann definierte Schnittstellen in entsprechenden Frameworks zur Verfügung.

3.4.5. Text-to-Speech Systeme



Als Text-to-Speech (TTS) Systeme werden Anwendungen bezeichnet, welche einen digitalen Text in digitale Audiodaten umwandeln. Diese Audiodaten können danach im Anschluss abgespielt werden, d. h. der Text wird mittels Software vorgelesen.

In den Anfangsjahren klangen die erzeugten Stimmen noch sehr künstlich. In den letzten Jahren konnte die Qualität der Ausgabe deutlich gesteigert werden. Einige Lösungen sind der menschlichen Aussprache schon sehr ähnlich. Allerdings fehlen immer noch die emotionalen Nuancen, welche Menschen erzeugen. TTS-Systeme sind heutzutage schon in viele digitale Geräte integriert und finden eine vielfache Verwendung, z. B. bei Siri, Alexa, Cortana oder verschiedenen Navigationssystemen. Außerdem helfen TTS-Systeme beispielsweise bei einer Sehbehinderung, mit Computern, Tablets, Smartphones oder anderen digitalen Geräten zu arbeiten.

Es gibt verschiedene TTS-Systeme auf dem Markt. Die einen können heruntergeladen, installiert und durch entsprechende Bibliotheken in das eigene Programm integriert werden. Andere System stehen als Cloud-Service zur Verfügung und können auf diese Weise in das Programm integriert werden. Allerdings ist dafür immer eine Internetverbindung notwendig. Beispiele für TTS-Systeme sind IBM Watson, Google Cloud TTS, Amazon Alexa TTS, Mircosoft Cortana, FreeTTS, iSpeech, Notevibes oder Natural Text Reader.

✓ Kompetenzcheck

1. Geben Sie an, welche Aussagen richtig sind.

   a. Windows 3.0 hatte als erstes Betriebssystem eine grafische Benutzeroberfläche.
   b. Textbox und Button sind GUI-Elemente.
   c. Es sollten so viele GUI-Elemente wie möglich auf einem Fenster platziert werden, um dem Benutzer eine kompakte Funktionalität zu ermöglichen.
   d. Unterstreichungen bei Textdarstellungen sollten nur für Links verwendet werden.
   e. Beim Corporate Design werden Farben verwendet, welche das Unternehmen widerspiegeln.
   f. Barrierefreiheit spielt heutzutage kaum noch eine Rolle bei den Designentscheidungen für Benutzeroberflächen.
   g. Für die Touchsteuerung von grafischen Benutzeroberflächen ist selten eine Umgestaltung der Oberfläche notwendig.
   h. Für eine rudimentäre Sprachsteuerung stehen KI-Services im Internet zur Verfügung.
   i. TTS-Systeme wandeln 2-D-Grafiken in 3-D-Grafiken um.
   j. IBM Watson ist ein TTS-System.

2. Stellen Sie Informationen zusammen, welche Erfahrungen Sie mit der Gestaltung von GUI gemacht haben und welche GUI-Elemente für welche Aufgaben von Ihnen bevorzugt eingesetzt werden. Begründen Sie Ihre Entscheidungen.

3. Bearbeiten Sie Kapitel 3 die Aufgaben 8 bis 10 im Handout und nutzen Sie die Aufgabe 11 für die Prüfungsvorbereitung.

4. Benutzerschnittstellen in einer Programmiersprache implementieren

➢ **Sie sollen die Benutzerschnittstellen implementieren, die Sie für Ihr Projekt geplant haben.**



Nachdem der Aufbau und die Inhalte der Benutzerschnittstellen der Anwendung feststehen, werden diese nun implementiert. Dabei können für eine Anwendung mit mehreren Benutzerschnittstellen durchaus unterschiedliche Technologien und Frameworks zum Einsatz kommen. Ein Firmenmitarbeiter, welcher am PC die Daten einer Anwendung pflegt, benötigt eine andere Schnittstelle als ein Kunde, welcher sich die Daten beispielsweise auf seinem Smartphone ansieht.

Der Schwerpunkt in diesem Kapitel liegt auf der Umsetzung von grafischen Benutzernutzeroberflächen. Dafür stehen viele Frameworks zur Verfügung, welche teilweise für bestimmte Plattformen und teilweise plattformübergreifend ausgelegt sind. Manche Programmiersprachen bringen eigene Frameworks mit, andere setzen auf externe Frameworks und haben eine entsprechende Anbindung geschaffen.

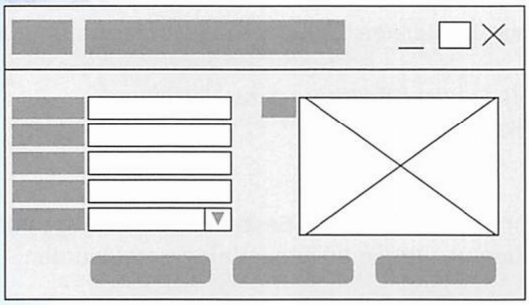
Wegen ihrer Vielfalt und den unzähligen Möglichkeiten, welche die heutigen Frameworks bieten, wird in diesem Kapitel nur ein Einblick in die GUI-Entwicklung gegeben. Zunächst werden die grundlegenden GUI-Elemente und deren Implementierung anhand eines abstrakten Beispiels vorgestellt. Mithilfe dieses Wissens wird dann exemplarisch nachfolgendes Beispiel umgesetzt.

Softwareprojekt „Yachthafen Resort“
Liegeplatzverwaltung

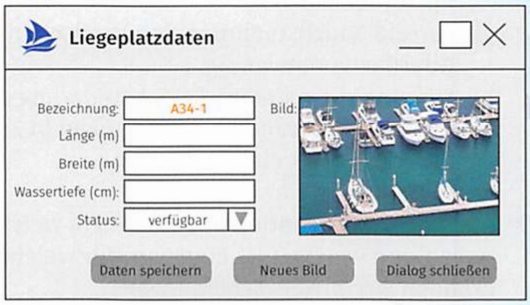
JIKU

Die Software soll u. a. ein Modul enthalten, mit dessen Hilfe die Liegeplätze des Yachthafen verwaltet werden können. In diesem Rahmen soll eine Benutzerschnittstelle geschaffen werden, mit deren Hilfe die Daten der einzelnen Liegeplätze eingegeben und notfalls geändert werden können.
Die Anforderungsspezifikation legt fest, dass folgende Daten für einen Liegeplatz erfasst werden sollen: Bezeichnung, Breite, Länge, Wassertiefe, ein Bild des Liegeplatzes.
Daneben soll der Status des Liegeplatzes angezeigt werden. Dieser kann verfügbar, vermietet, belegt oder gesperrt sein. Außerdem soll ein Corporate Design mit den Farben und dem Logo der Yachthafenresidenz gewählt werden.
Auf Grundlage dieser Vorgaben wurden schon ein Wireframe und ein Mockup erstellt. Diese sollen als Grundlage für die programmiertechnische Umsetzung dieser Benutzerschnittstelle dienen.

Wireframe



Mockup



Zum Status des Liegeplatzes siehe auch Zustandsdiagramm, Kapitel 2.1.2, S. 119.

4.1. Bibliotheken und Frameworks auswählen

- **Sie sollen aus verschiedenen Möglichkeiten eine passende GUI-Bibliothek oder ein passendes GUI-Framework auswählen.**

Bevor die grafische Benutzeroberfläche implementiert werden kann, muss feststehen, welche Programmiersprache und welches Framework benutzt werden soll. In vielen Fällen steht das bereits im Vorfeld fest, z. B. wenn eine Software erweitert werden soll. In diesem Fall werden schon eine bestimmte Programmiersprache und ein GUI-Framework verwendet, mit denen in der Regel weiter programmiert wird. In anderen Fällen ist die Programmiersprache schon vorgegeben, z. B. weil diese die Entwicklungssprache für dieses Projekt ist, aber das Framework für die Oberflächenentwicklung steht noch nicht fest. Und auch der umgekehrte Fall ist denkbar: Das Framework ist zwingend vorgegeben, z. B. soll das plattformübergreifende QT-Framework eingesetzt werden, allerdings ist die Programmiersprache nicht festgelegt. Oder es stehen weder die Programmiersprache noch das Framework fest. Wenn die Werkzeuge für das Projekt noch nicht feststehen, steht man vor der Frage, welche Programmiersprache und welches Framework man auswählt (siehe auch Jahrgangsband 1, Modul 5). Im Folgenden wird die Auswahl eines GUI- Framework näher betrachtet. Viele der nachfolgend angesprochenen Punkte lassen sich aber auch für die Auswahl anderer Arten von Frameworks heranziehen. Zunächst werden hier einige grundlegende Begriffe definiert.

Grundlegende Begriffe	
Bibliothek	Eine Bibliothek ist eine Sammlung von Klassen, Modulen oder Funktionen, welche schon zur Lösung von einem bestimmten Problem implementiert wurden. Bibliotheken können in den eigenen Code eingebunden und genutzt werden. Zum Beispiel wird eine Bitmap-Verarbeitungsbibliothek, welche Funktionen zum Laden und Bearbeiten von Bitmap-Bildern anbietet, im Programm eingebunden. Somit braucht dieser Teil nicht mehr implementiert werden. Normalerweise bieten Bibliotheken nur einen abgegrenzten Funktionsbereich an, z. B. die Verarbeitung von Bildern oder das Bearbeiten von Zip-Dateien. Jede Bibliothek besitzt eine API, über die auf die Funktionalität zugegriffen werden kann.
Framework	Ein Framework ist eine große Bibliothek oder eine Gruppe von Bibliotheken, welche verschiedene Dienste bereitstellen und nicht nur einen Dienst wie eine Bibliothek. Zum Beispiel bietet das .Net-Framework von Microsoft eine Vielzahl von Funktionalitäten an, welche man zur Entwicklung einer Anwendung benötigt. Frameworks bieten häufig eine vollständige Grundstruktur für eine Anwendung. Auf Grundlage dieses Rahmens kann dann der eigene Code erstellt werden.
Application Programming Interface (API)	Eine API ist die Schnittstelle, mit deren Hilfe die Funktionalität einer Bibliothek oder eines Frameworks genutzt werden kann. Dazu stellt die API verschiedene Funktionen bereit, die im eigenen Programm eingebaut werden können und somit die Interaktion mit der Bibliothek oder dem Framework ermöglichen.

Wenn man die Möglichkeit hat, ein GUI-Framework für seine Anwendung auszuwählen, dann sollte dieses nach bestimmten Kriterien erfolgen und nicht aus dem Bauch heraus. Anhand eines Kriterienkataloges kann entschieden werden, ob ein Framework zur Lösung der gestellten Anforderungen überhaupt geeignet ist.

Kriterien zur Auswahl eines GUI-Frameworks

Abdeckung der Anforderungen
Zunächst gilt es festzustellen, ob sich mit dem Framework alle Anforderungen umsetzen lassen und ob es sich auch eventuell für zukünftige Erweiterungen einsetzen lässt. Wenn eine grafische Benutzerschnittstelle für den Desktopbereich entwickelt werden soll, könnte trotzdem in die Überlegungen einfließen, ob das Framework auch die Entwicklung von Oberflächen für den mobilen Bereich ermöglicht. Außerdem sollte überprüft werden, welche Entwicklungsbereiche noch durch das Framework mit Funktionalität abgedeckt werden, z. B. der Bereich der Informationssicherheit.

Popularität, Unterstützung und Dokumentation
Je bekannter und anerkannter das Framework ist, desto mehr wird es gepflegt, weiterentwickelt und vervollständigt. Ein weiteres Kriterium, das nicht unterschätzt werden sollte, ist, wie leicht man Antworten auf Fragen finden und Hilfe erhalten kann. Dieser Bereich betrifft z. B. den verfügbaren Support und Möglichkeiten zur Schulung. Aber auch eine große Community kann sehr hilfreich sein. Eine gute Dokumentation ist eine absolute Notwendigkeit. Ein gut dokumentiertes Werkzeug ist einfacher zu verstehen und einzusetzen.

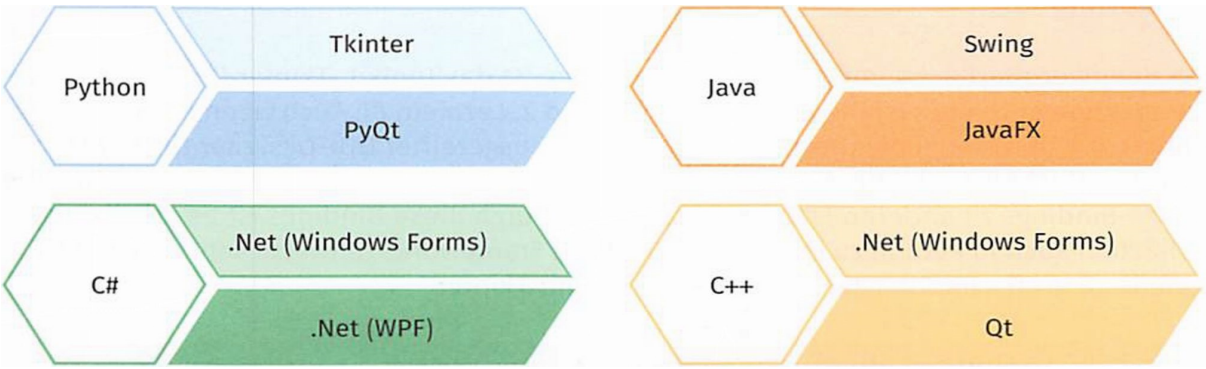
Lizenz und Kosten
Ein weiterer wichtiger Punkt ist die Frage nach der Lizenz, unter der mit dem Framework entwickelt werden kann. Auch eventuelle Lizenzkosten sind von Belang. Das Letzte, was man möchte, ist nach der Erstellung der Anwendung herauszufinden, dass die Lizenz es nicht erlaubt, die Anwendung kommerziell zu vertreiben. Das gilt auch für zusätzliche Plug-in, welche für die Entwicklung verwendet werden.

Entwurfsmuster
Heutzutage verwendet fast jedes GUI-Framework das MVC-Pattern (siehe auch S. 27). Allerdings gibt es dort verschiedene Varianten, z. B. das MVP-Pattern oder das MVVM-Pattern. Falls das konkrete Entwurfsmuster für das Projekt eine Rolle spielt, sollte dieser Aspekt der Auswahl der Frameworks berücksichtigt werden.

Installation
Auch wenn dieser Punkt in der Regel unterschätzt wird, spielt die Einfachheit der Installation eine sehr wichtige Rolle. In einigen Fällen können die Installation sowie das Einstellen der richtigen Werte in der Konfigurationsdaten für spätere Administratoren eine sehr zeitaufwendige, komplizierte Angelegenheit sein. Umso unkomplizierter diese Punkte sind, umso besser ist das Framework zu bewerten.

Erlernbarkeit
Jedes Framework besitzt sein eigenes Universum aus Namenskonventionen, Verzeichnisstrukturen und Vorgehensweisen. Diese müssen von dem Entwickler verstanden und erlernt werden. Umso mehr sich das Framework an bekannte Konventionen hält, umso schneller lässt sich der Gebrauch des Frameworks erlernen. Eine einfache Erlernbarkeit ist insofern wichtig, weil man als Entwickler ein Problem lösen möchte und nicht aufwendig ein Framework erlernen will.

In der Regel hat man nicht die Möglichkeit, sich die Programmiersprache oder das Framework auszusuchen, weil diese Parameter schon durch die Rahmenbedingungen des Projektes vorgegeben sind. Wenn doch eine Auswahlmöglichkeit besteht, dann steht in vielen Fällen die Programmiersprache bereits fest und es wird ein GUI-Framework gesucht, welches auch mit dieser Programmiersprache entwickelt werden kann. Nachfolgend sind einige Beispiele



Für den Webbereich erleichtern wiederum Web-Frameworks das Entwickeln von Web-Anwendungen. Auch hier gibt es eine große Auswahl an Frameworks, welche einer sehr dynamischen Entwicklung unterliegen.



✓ Kompetenzcheck

1. Geben Sie an, welche Aussagen richtig sind.

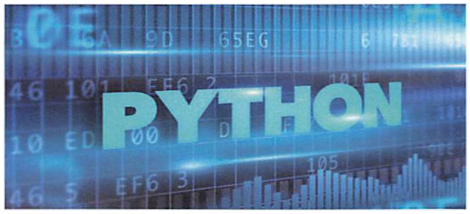
   a. Eine Bibliothek setzt sich aus mehreren Frameworks zusammen.
   b. Ein Framework stellt in der Regel Benutzern mehrere Funktionalitäten zur Verfügung.
   c. Eine API ist eine spezielle Bibliothek, welche genau auf die Bedürfnisse der Benutzer zugeschnitten ist.
   d. Lizenzreglungen spielen bei der Auswahl von Frameworks eine untergeordnete Rolle.
   e. Zu jeder Programmiersprache gibt es genau ein GUI-Framework.
   f. Das Swing-Framework ist ein GUI-Framework für Python.
   g. Das Qt-Framework kann u.a. in C++ und in Python verwendet werden.
   h. Angular ist ein Web-Framework, welches im Zusammenhang mit der Programmiersprache JavaScript verwendet werden kann.

2. Bearbeiten Sie im Handout die Aufgabe 1 des Kapitels 4.

4.2. Benutzerschnittstellen in Python realisieren

➢ **Sie sollen auf der Grundlage von Wireframes und Mockups grafische Benutzerschnittstellen (GUI) in Python realisieren.**

Die Programmierung einer GUI in Python ist sehr komplex und vielfältig (siehe auch Jahrgangsband 2, Modul 8). Deswegen werden in diesem Kapitel nur die grundlegendsten GUI-Elemente angesprochen. Durch das Entwickeln von verschiedenen Oberflächen mit unterschiedlichen Anforderungen vertieft und erweitert sich mit der Zeit das Wissen über die einzelnen GUI- Elemente und die GUI-Programmierung in Python im Allgemeinen.

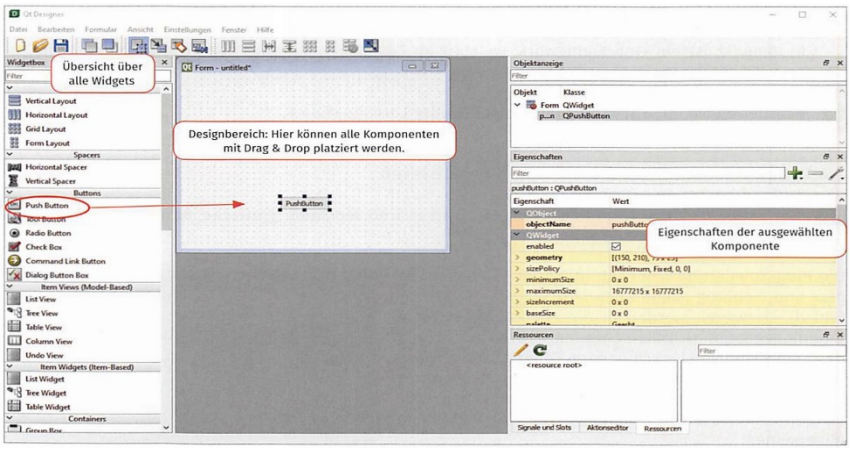


4.2.1. GUI-Programmierung

Um eine grafische Benutzeroberfläche unter Python zu erstellen, ist das Toolkit ,,Tkinter (Tk)" in der Standardbibliothek von Python enthalten (siehe auch Jahrgangsband 2, Modul 8). Auch wenn es verschiedene Entwicklungsansätze gibt, steht bisher (April 2022) noch kein ausgereifter GUI-Designer für Tkinter zur Verfügung, sodass Oberflächen noch immer per Hand programmiert werden müssen. Allerdings besitzt Python verschiedene Bindings zu anderen Frameworks, z. B. Qt. Durch diese Bindings ist es möglich, entsprechende Oberflächen auch in Python zu entwickeln. Da diese Frameworks teilweise sehr mächtig sind, lassen sich ausgefeiltere grafische Oberflächen gestalten als mit Tkinter.

Grundlegende GUI-Elemente in Tkinter und PyQt		
	Tkinter	**PyQt**
Fenster	Frame	QWidget
Tabs	Notebook, Frame	QTabWidget, QWidget
Menü	Menu	QMenuBar
Textbox	Entry	QLineEdit
Label	Label	QLabel
Button	Button	QPushButton
Radiobutton	Radiobutton	QRadioButton
Checkbox	Checkbutton	QCheckBox
Listbox	Listbox	QListWidget
Combobox	Combobox	QComboBox
Bild	PhotoImage	QPixmap
Schieberegler	Scale	QSlider

Außerdem können, wenn vorhanden, die Werkzeuge dieser Frameworks mitgenutzt werden. Im Falle von Qt ist das der Qt-Designer.



Der QT-Designer entspricht in seinem Grundaufbau und seiner Funktionalität allen anderen bekannten GUI-Designern, z. B. dem GUI-Builder in Java (siehe auch Jahrgangsband 2, Modul 8). Die Anordnung und die Gestaltung der einzelnen Bereiche mag unterschiedlich sein, aber man findet sich sehr schnell zurecht, wenn man das Grundprinzip eines GUI-Designers verstanden hat. Da es sich bei Qt um ein externes Framework handelt, muss dieses zunächst auf dem System und in PyCharm installiert werden. Dazu sind unter Windows folgende Schritte notwendig:

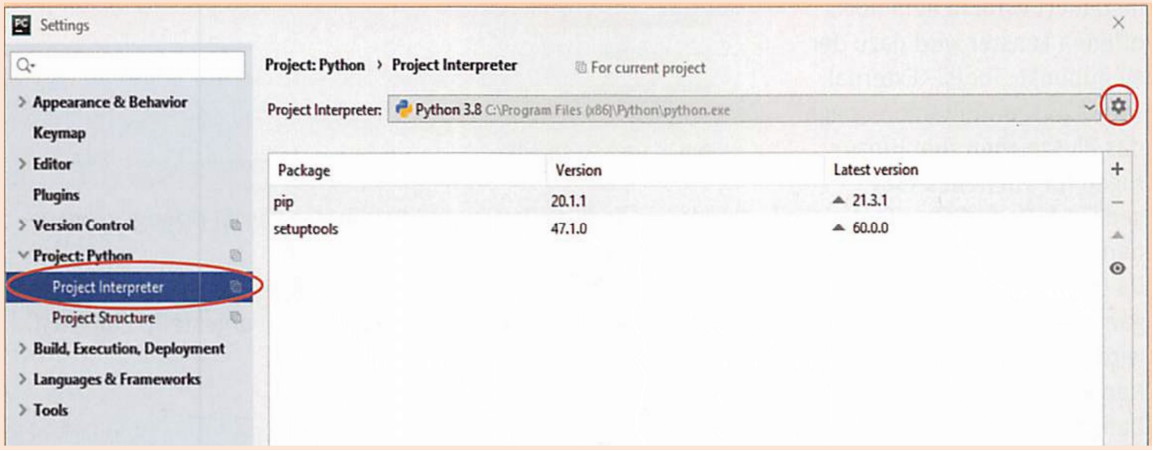
Installieren von Qt und dem QT-Designer für Python in PyCharm

1 Installieren von pyqt6 und pyqt6-tools

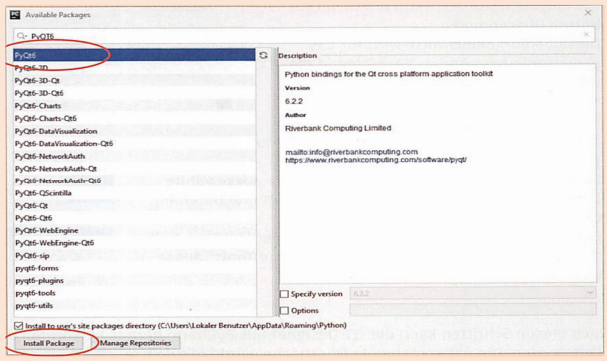
Dazu wird in einem Konsolenfenster folgende Anweisung ausgeführt: pip install pyqt6 pyqt6-tools.
Damit werden alle notwendigen Dateien auf dem System installiert.

2 Installieren von dem Package PyQt6 in PyCharm

Als erstes wird dazu in PyCharm ein neues Projekt angelegt. Danach wird unter dem Menüpunkt „File" der Untermenüpunkt „Settings..." aufgerufen und es erscheint folgendes Fenster.



Hier wird nun der Menüpunkt „Projektinterpreter" ausgewählt und das Pluszeichen gedrückt. Im dem darauf sich öffnenden Fenster wird nach PyQt6 gesucht, dieses Package ausgewählt und durch Drücken des Buttons „Install Package" installiert. Danach kann das Fenster wieder geschlossen werden.



Damit steht das Package PYQt6 nun in PyCharm zur Verfügung.

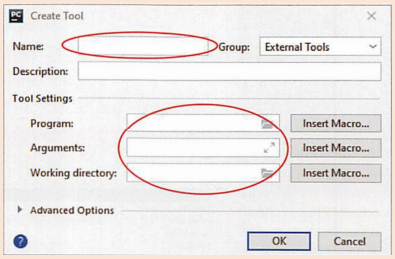
3 Installieren des QT- Designers und pyuic

Um den QT-Designer in PyCharm nutzen zu können, muss dieser als externes Tool installiert werden. Damit können dann die Oberflächen entwickelt und zunächst einmal als ui-Datei abgespeichert werden. Damit diese ui-Datei in Python genutzt werden kann, muss sie anschließend konvertiert werden. Dazu dient das pyuic-Tool, welches auch als externes Tool installiert wird. In dem noch offenen Fenster wird dazu der Menüpunkt „Tools - External Tools" ausgewählt und danach das Pluszeichen zum Hinzufügen für ein neues Tool gedrückt. Daraufhin öffnet sich ein Fenster, in dem die Daten und Parameter des Werkzeugs eingegeben werden, welche in PyCharm eingebunden werden sollen.

Da der QT-Designer und pyuic eingebunden werden sollen, muss der nachfolgend beschriebene Vorgang zweimal durchgeführt werden. Exemplarisch wird es für den QT-Designer vorgestellt. Zunächst wird ein frei wählbarer Name für das Werkzeug vergeben.

Name: QT-Designer

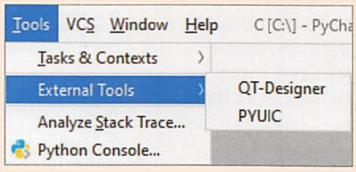
Danach wird eingeben, wo das Programm zu finden ist und das Arbeitsverzeichnis. Für den QT-Designer sind das folgende Werte:



Program:PATH_ZUR_EXE\designer.exe
Working directory: $ProjectFileDir$

Die Eingaben werden dann mit dem OK- Button übernommen. Dieselbe Vorgehensweise wird für das pyuic-Tool angewendet. Nachfolgend die einzugebenden Daten:

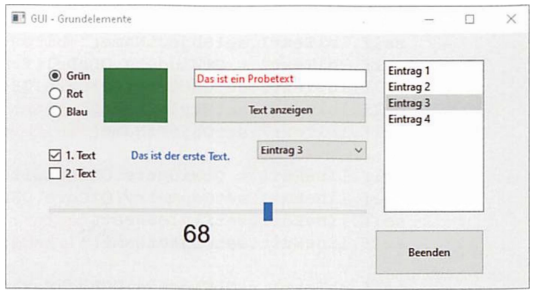
Name: PYUIC
Program: PATH_ZUR_EXE\pyuic6.exe
Arguments: $FileName$ -o $FileNameWithoutExtension$.py
Working directory: $ProjectFileDir$



Bei den Argumenten kommt hinzu, dass die ausgewählte Datei mit der Endung „ui" in eine gleichlautende Datei mit der Endung „py" konvertiert wird.

Danach können alle Fenster geschlossen werden. Nun sollten die beiden Werkzeuge unter „Tools - Externe Tools" ausgewählt werden können.

Nach diesen Schritten kann der QT-Designer aus PyCharm heraus aufgerufen und eine Oberfläche entwickelt werden. Die nebenstehende Oberfläche wurde mit dem QT-Designer entwickelt und danach in eine Python-Datei konvertiert. In diesem Beispiel sind grundlegende GUI-Elemente dargestellt und es wurde eine einfache Funktionalität hinterlegt. Damit soll das Prinzip der Erstellung und des Event- Handlings des QT-Frameworks kompakt dargestellt werden. Nachfolgend ist der Quellcode aufgeführt und kommentiert, welcher vom QT-Designer erstellt wurde. Dieser kann als Grundlage genommen werden, für den Fall, dass man doch das eine oder andere Element per Hand erstellen und initialisieren möchte.



Quelltext, welcher vom QT-Designer erstellt wurde

```
# Form implementation generated from reading ui file ,bsp_gui_elemente_
frmMain.ui'
#
# Created by: PyQt6 UI code generator 6.2.2
#
# WARNING: Any manual changes made to this file will be lost when pyuic6

# is run again.  Do not edit this file unless you know what you are doing.

from PyQt6 import QtCore, QtGui, QtWidgets

class Ui_frmMain(object):
    def setupUi(self, frmMain):
        frmMain.setObjectName("frmMain")
        frmMain.resize(584, 289)

        self.btnClose = QtWidgets.QPushButton(frmMain)
        self.btnClose.setGeometry(QtCore.QRect(410, 220, 121, 51))
        self.btnClose.setObjectName("btnClose")
        self.btnShow = QtWidgets.QPushButton(frmMain)
        self.btnShow.setGeometry(QtCore.QRect(210, 70, 191, 31))
        self.btnShow.setObjectName("btnShow")

        self.rbGreen = QtWidgets.QRadioButton(frmMain)
        self.rbGreen.setGeometry(QtCore.QRect(50, 40, 82, 17))
        self.rbGreen.setChecked(True)
        self.rbGreen.setObjectName("rbGreen")
        self.rbRed = QtWidgets.QRadioButton(frmMain)
        self.rbRed.setGeometry(QtCore.QRect(50, 60, 82, 17))
        self.rbRed.setObjectName("rbRed")
        self.rbBlue = QtWidgets.QRadioButton(frmMain)
        self.rbBlue.setGeometry(QtCore.QRect(50, 80, 82, 17))
        self.rbBlue.setObjectName("rbBlue")

        self.lblColor = QtWidgets.QLabel(frmMain)
        self.lblColor.setGeometry(QtCore.QRect(110, 40, 71, 61))
        self.lblColor.setStyleSheet("background-color: rgb(0, 170, 0);")
        self.lblColor.setText("")
        self.lblColor.setObjectName("lblColor")
        self.lblText1 = QtWidgets.QLabel(frmMain)
        self.lblText1.setGeometry(QtCore.QRect(140, 130, 121, 16))
```

Anlegen und Initialisieren des Fensters

Anlegen und Initialisieren der Buttons

Anlegen und Initialisieren der Radiobuttons

Anlegen und Initialisieren der Labels

```
self.lblText1.setStyleSheet("color: rgb(0, 170, 255);")
self.lblText1.setObjectName("lblText1")
self.lblText2 = QtWidgets.QLabel(frmMain)
self.lblText2.setGeometry(QtCore.QRect(140, 150, 121, 16))
self.lblText2.setStyleSheet("color: rgb(0, 170, 0);")
self.lblText2.setObjectName("lblText2")

self.lineEdit = QtWidgets.QLineEdit(frmMain)
self.lineEdit.setGeometry(QtCore.QRect(210, 40, 191, 20))
self.lineEdit.setStyleSheet("color: rgb(255, 0, 127);")
self.lineEdit.setObjectName("lineEdit")

self.cbText1 = QtWidgets.QCheckBox(frmMain)
self.cbText1.setGeometry(QtCore.QRect(50, 130, 70, 17))
self.cbText1.setChecked(True)
self.cbText1.setObjectName("cbText1")
self.cbText2 = QtWidgets.QCheckBox(frmMain)
self.cbText2.setGeometry(QtCore.QRect(50, 150, 70, 17))
self.cbText2.setChecked(True)
self.cbText2.setObjectName("cbText2")

self.hSlider = QtWidgets.QSlider(frmMain)
self.hSlider.setGeometry(QtCore.QRect(50, 190, 351, 22))
self.hSlider.setOrientation(QtCore.Qt.Orientation.Horizontal)
self.hSlider.setObjectName("hSlider")
self.lblSliderValue = QtWidgets.QLabel(frmMain)
self.lblSliderValue.setGeometry(QtCore.QRect(190, 210, 47, 31))
self.lblSliderValue.setStyleSheet("font: 75 20pt \"MS Shell Dlg
                                  2\";")
self.lblSliderValue.setAlignment(QtCore.Qt.AlignmentFlag.
                                        AlignCenter)
self.lblSliderValue.setObjectName("lblSliderValue")

self.listWidget = QtWidgets.QListWidget(frmMain)
self.listWidget.setGeometry(QtCore.QRect(420, 30, 111, 171))
self.listWidget.setObjectName("listWidget")
item = QtWidgets.QListWidgetItem()
self.listWidget.addItem(item)
item = QtWidgets.QListWidgetItem()
self.listWidget.addItem(item)
item = QtWidgets.QListWidgetItem()
self.listWidget.addItem(item)
item = QtWidgets.QListWidgetItem()
self.listWidget.addItem(item)

self.comboBox = QtWidgets.QComboBox(frmMain)
self.comboBox.setGeometry(QtCore.QRect(280, 120, 121, 22))
self.comboBox.setObjectName("comboBox")
self.comboBox.addItem("")
self.comboBox.addItem("")
self.comboBox.addItem("")

self.retranslateUi(frmMain)
self.btnClose.clicked.connect(self.btnClose_click)
self.btnShow.clicked.connect(self.btnShow_click)
self.rbGreen.clicked.connect(self.rbGreen_click)
self.rbRed.clicked.connect(self.rbRed_click)
self.rbBlue.clicked.connect(self.rbBlue_click)
```

Anlegen und Initialisieren der Textbox

Anlegen und Initialisieren der Checkboxen

Anlegen und Initialisieren des Schiebereglers

Anlegen und Initialisieren der Listbox mit verschiedenen Einträgen

Anlegen und Initialisieren der Combobox mit verschiedenen Einträgen

Definieren der Events und zuordnen von Methoden für einzelne GUI-Elemente. Die Methoden müssen dann im Hauptprogramm implementiert werden.

```
        self.cbText1.clicked.connect(self.cbText1_click)
        self.cbText2.clicked.connect(self.cbText2_click)
        self.hSlider.sliderMoved['int'].connect(self.hSlider_moved)
        QtCore.QMetaObject.connectSlotsByName(frmMain)

    def retranslateUi(self, frmMain):
        _translate = QtCore.QCoreApplication.translate
        frmMain.setWindowTitle(_translate("frmMain",
                                          "GUI - Grundelemente"))
        self.btnClose.setText(_translate("frmMain", "Beenden"))
        self.rbGreen.setText(_translate("frmMain", "Grün"))
        self.rbRed.setText(_translate("frmMain", "Rot"))
        self.rbBlue.setText(_translate("frmMain", "Blau"))
        self.btnShow.setText(_translate("frmMain", "Text anzeigen"))
        self.cbText1.setText(_translate("frmMain", "1. Text"))
        self.cbText2.setText(_translate("frmMain", "2. Text"))
        self.lblText1.setText(_translate("frmMain",
                                         "Das ist der erste Text."))
        self.lblText2.setText(_translate("frmMain",
                                         "Das ist der zweite Text."))
        self.lblSliderValue.setText(_translate("frmMain", "0"))
        __sortingEnabled = self.listWidget.isSortingEnabled()
        self.listWidget.setSortingEnabled(False)
        item = self.listWidget.item(0)
        item.setText(_translate("frmMain", "Eintrag 1"))
        item = self.listWidget.item(1)
        item.setText(_translate("frmMain", "Eintrag 2"))
        item = self.listWidget.item(2)
        item.setText(_translate("frmMain", "Eintrag 3"))
        item = self.listWidget.item(3)
        item.setText(_translate("frmMain", "Eintrag 4"))
        self.listWidget.setSortingEnabled(__sortingEnabled)
        self.comboBox.setCurrentText(_translate("frmMain", "Eintrag 1"))
        self.comboBox.setItemText(0, _translate("frmMain", "Eintrag 1"))
        self.comboBox.setItemText(1, _translate("frmMain", "Eintrag 2"))
        self.comboBox.setItemText(2, _translate("frmMain", "Eintrag 3"))
```

Um die Oberfläche nutzen zu können, muss noch ein Hauptprogramm geschrieben werden, in dem die Oberfläche erzeugt und ausgeführt wird. Außerdem sind hier noch die Event-Methoden der einzelnen GUI-Elemente mit Leben zu füllen. Nachfolgend ist der Quellcode für das Hauptprogramm aufgeführt und kommentiert.

Quellcode des Hauptprogramms

```
from bsp_gui_elemente_frmMain import Ui_frmMain
from PyQt6 import QtWidgets
from PyQt6.QtWidgets import QApplication, QMessageBox
import sys

class Bsp_Form(QtWidgets.QWidget,Ui_frmMain):
    def __init__(self):
        super(Bsp_Form,self).__init__()
        self.setupUi(self)

    def btnClose_click(self):
        app.quit()
```

Einbinden entsprechender Module und des GUI-Fensters

Anlegen und Initialisieren einer Klasse, welche u. a. von der entwickelten Oberfläche abgeleitet wird. Hier werden nun die Event-Methoden implementiert.

```
    def btnShow_click(self):
        text = self.lineEdit.text()
        QMessageBox.information(self, 'Information', text)

    def rbGreen_click(self):
        self.lblColor.setStyleSheet("background-color: rgb(0, 170, 0);")

    def rbRed_click(self):
        self.lblColor.setStyleSheet("background-color: rgb(255, 0, 0);")

    def rbBlue_click(self):
        self.lblColor.setStyleSheet("background-color: rgb(0, 0, 255);")

    def cbText1_click(self):
        if (self.cbText1.isChecked()):
            self.lblText1.setVisible(True)
        else:
            self.lblText1.setVisible(False)

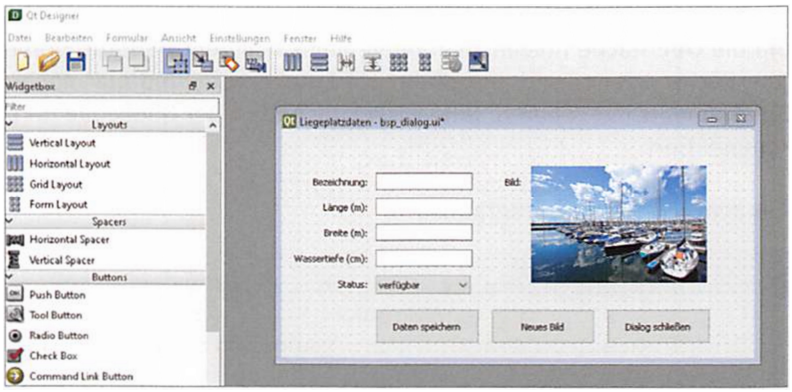
    def cbText2_click(self):
        if (self.cbText2.isChecked()):
            self.lblText2.setVisible(True)
        else:
            self.lblText2.setVisible(False)

    def hSlider_moved(self):
        self.lblSliderValue.setText(str(self.hSlider.value()))


if __name__ == '__main__':
    app = QApplication(sys.argv)
    form = Bsp_Form()
    form.show()
    sys.exit(app.exec())
```

Erstellen und Anzeigen der Oberfläche und Ausführen des Programms.

4.2.2. Umsetzung der Beispielaufgabe



Die Oberfläche für die Beispielaufgabe wird wieder mit dem QT-Designer entwickelt. Nachdem alle GUI-Elemente auf der Oberfläche angeordnet sind und die entsprechenden Eigenschaften gesetzt wurden, können nun die Events, z. B. Buttonklicks, definiert und implementiert werden. Um die Events festzulegen, wird ebenfalls der QT-Designer verwendet. Dazu wird auf den Menüpunkt „Sig nale und Slots bearbeiten" gewechselt. Wenn ein Event ausgewählt wurde, wird der entsprechende Event-Code (Funktionsname) automatisch generiert. Wie schon zuvor erwähnt, müssen die Funktionen noch angelegt und implementiert werden. In diesem Beispiel wurde auf Implementierung der Funktionalität verzichtet, weil es hier vor allem um die Erstellung der Benutzeroberfläche geht. Das Thema der Implementierung der Funktionalität wird im nächsten Lernfeld thematisiert. Nachfolgend sind der vom QT-Designer erzeugte Quelltext und das notwendige Hauptprogramm aufgelistet und kommentiert.

Quelltext, welcher vom QT-Designer erstellt wurde

```
# Form implementation generated from reading ui file ,bsp_dialog.ui'
#
# Created by: PyQt6 UI code generator 6.2.2
#
# WARNING: Any manual changes made to this file will be lost when pyuic6
# is run again.  Do not edit this file unless you know what you are doing.

from PyQt6 import QtCore, QtGui, QtWidgets

class Ui_frmLiegeplatzerfasssung(object):
    def setupUi(self, frmLiegeplatzerfasssung):
        frmLiegeplatzerfasssung.setObjectName("frmLiegeplatzerfasssung")
        frmLiegeplatzerfasssung.resize(552, 268)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("C:/Bild1.png"),
                       QtGui.QIcon.Mode.Normal, QtGui.QIcon.State.On)
        frmLiegeplatzerfasssung.setWindowIcon(icon)
        self.txtBezeichnung = QtWidgets.QLineEdit(frmLiegeplatzerfasssung)
        self.txtBezeichnung.setGeometry(QtCore.QRect(110, 50, 113, 20))
        self.txtBezeichnung.setStyleSheet("color: rgb(170, 85, 0);")
        self.txtBezeichnung.setAlignment(QtCore.Qt.AlignmentFlag.
                                                        AlignCenter)
        self.txtBezeichnung.setObjectName("txtBezeichnung")
        self.lblBezeicnung = QtWidgets.QLabel(frmLiegeplatzerfasssung)
        self.lblBezeicnung.setGeometry(QtCore.QRect(30, 50, 71, 20))
        self.lblBezeicnung.setAlignment(QtCore.Qt.AlignmentFlag.
                                                         AlignRight|
                                        QtCore.Qt.AlignmentFlag.
                                                       AlignTrailing|
                                        QtCore.Qt.AlignmentFlag.
                                                        AlignVCenter)
        self.lblBezeicnung.setObjectName("lblBezeicnung")
        self.txtLaenge = QtWidgets.QLineEdit(frmLiegeplatzerfasssung)
        self.txtLaenge.setGeometry(QtCore.QRect(110, 80, 113, 20))
        self.txtLaenge.setObjectName("txtLaenge")
        self.lblLaenge = QtWidgets.QLabel(frmLiegeplatzerfasssung)
        self.lblLaenge.setGeometry(QtCore.QRect(30, 80, 71, 20))
        self.lblLaenge.setAlignment(QtCore.Qt.AlignmentFlag.AlignRight|
                                    QtCore.Qt.AlignmentFlag.AlignTrailing|
                                    QtCore.Qt.AlignmentFlag.AlignVCenter)
        self.lblLaenge.setObjectName("lblLaenge")
        self.txtBreite = QtWidgets.QLineEdit(frmLiegeplatzerfasssung)
        self.txtBreite.setGeometry(QtCore.QRect(110, 110, 113, 20))
        self.txtBreite.setObjectName("txtBreite")
        self.lblBreite = QtWidgets.QLabel(frmLiegeplatzerfasssung)
        self.lblBreite.setGeometry(QtCore.QRect(30, 110, 71, 20))
        self.lblBreite.setAlignment(QtCore.Qt.AlignmentFlag.AlignRight|
                                    QtCore.Qt.AlignmentFlag.AlignTrailing|
                                    QtCore.Qt.AlignmentFlag.AlignVCenter)
        self.lblBreite.setObjectName("lblBreite")
        self.txtWassertiefe = QtWidgets.QLineEdit(frmLiegeplatzerfasssung)
        self.txtWassertiefe.setGeometry(QtCore.QRect(110, 140, 113, 20))
        self.txtWassertiefe.setObjectName("txtWassertiefe")
        self.lblWasserteife = QtWidgets.QLabel(frmLiegeplatzerfasssung)
        self.lblWasserteife.setGeometry(QtCore.QRect(0, 140, 101, 20))
```

```
        self.lblWasserteife.setAlignment(QtCore.Qt.AlignmentFlag.
                                                        AlignRight|
                                         QtCore.Qt.AlignmentFlag.
                                                      AlignTrailing|
                                         QtCore.Qt.AlignmentFlag.
                                                       AlignVCenter)
        self.lblWasserteife.setObjectName("lblWasserteife")
        self.lblBild = QtWidgets.QLabel(frmLiegeplatzerfasssung)
        self.lblBild.setGeometry(QtCore.QRect(240, 50, 41, 20))
        self.lblBild.setAlignment(QtCore.Qt.AlignmentFlag.AlignRight|
                                  QtCore.Qt.AlignmentFlag.AlignTrailing|
                                  QtCore.Qt.AlignmentFlag.AlignVCenter)
        self.lblBild.setObjectName("lblBild")
        self.btnSave = QtWidgets.QPushButton(frmLiegeplatzerfasssung)
        self.btnSave.setGeometry(QtCore.QRect(110, 210, 121, 41))
        self.btnSave.setStyleSheet("")
        self.btnSave.setObjectName("btnSave")
        self.btnNew = QtWidgets.QPushButton(frmLiegeplatzerfasssung)
        self.btnNew.setGeometry(QtCore.QRect(245, 210, 121, 41))
        self.btnNew.setObjectName("btnNew")
        self.btnClose = QtWidgets.QPushButton(frmLiegeplatzerfasssung)
        self.btnClose.setGeometry(QtCore.QRect(380, 210, 121, 41))
        self.btnClose.setObjectName("btnClose")
        self.label = QtWidgets.QLabel(frmLiegeplatzerfasssung)
        self.label.setGeometry(QtCore.QRect(290, 50, 211, 141))
        self.label.setStyleSheet("")
        self.label.setText("")
        self.label.setPixmap(QtGui.QPixmap("../../Bilder/Liegeplatz.jpg"))
        self.label.setScaledContents(True)
        self.label.setObjectName("label")
        self.cbStatus = QtWidgets.QComboBox(frmLiegeplatzerfasssung)
        self.cbStatus.setGeometry(QtCore.QRect(110, 170, 111, 22))
        self.cbStatus.setLayoutDirection(QtCore.Qt.LayoutDirection.
                                                         LeftToRight)
        self.cbStatus.setObjectName("cbStatus")
        self.cbStatus.addItem("")
        self.cbStatus.addItem("")
        self.cbStatus.addItem("")
        self.cbStatus.addItem("")
        self.lblStatus = QtWidgets.QLabel(frmLiegeplatzerfasssung)
        self.lblStatus.setGeometry(QtCore.QRect(0, 170, 101, 20))
        self.lblStatus.setAlignment(QtCore.Qt.AlignmentFlag.AlignRight|
                                    QtCore.Qt.AlignmentFlag.AlignTrailing|
                                    QtCore.Qt.AlignmentFlag.AlignVCenter)
        self.lblStatus.setObjectName("lblStatus")

        self.retranslateUi(frmLiegeplatzerfasssung)
        self.btnSave.clicked.connect(self.btnSave_click)
        self.btnNew.clicked.connect(self.btnNew_click)
        self.btnClose.clicked.connect(self.btnClose_click)
        QtCore.QMetaObject.connectSlotsByName(frmLiegeplatzerfasssung)

    def retranslateUi(self, frmLiegeplatzerfasssung):
        _translate = QtCore.QCoreApplication.translate
        frmLiegeplatzerfasssung.setWindowTitle(_translate(
                            "frmLiegeplatzerfasssung", "Liegeplatzdaten"))
        self.lblBezeicnung.setText(_translate("frmLiegeplatzerfasssung",
                                                        "Bezeichnung:"))
```

```
        self.lblLaenge.setText(_translate("frmLiegeplatzerfasssung",
                                          "Länge (m):"))
        self.lblBreite.setText(_translate("frmLiegeplatzerfasssung",
                                          "Breite (m):"))
        self.lblWasserteife.setText(_translate("frmLiegeplatzerfasssung",
                                               "Wassertiefe (cm):"))
        self.lblBild.setText(_translate("frmLiegeplatzerfasssung",
                                        "Bild:"))
        self.btnSave.setText(_translate("frmLiegeplatzerfasssung",
                                        "Daten speichern"))
        self.btnNew.setText(_translate("frmLiegeplatzerfasssung",
                                       "Neues Bild"))
        self.btnClose.setText(_translate("frmLiegeplatzerfasssung",
                                         "Dialog schließen"))
        self.cbStatus.setItemText(0, _translate("frmLiegeplatzerfasssung",
                                                "verfügbar"))
        self.cbStatus.setItemText(1, _translate("frmLiegeplatzerfasssung",
                                                "vermietet"))
        self.cbStatus.setItemText(2, _translate("frmLiegeplatzerfasssung",
                                                "belegt"))
        self.cbStatus.setItemText(3, _translate("frmLiegeplatzerfasssung",
                                                "gesperrt"))
        self.lblStatus.setText(_translate("frmLiegeplatzerfasssung",
                                          "Status:"))
```

Quellcode des Hauptprogramms

```
from bsp_dialog import Ui_frmLiegeplatzerfasssung
from PyQt6 import QtWidgets
from PyQt6.QtWidgets import QApplication
import sys

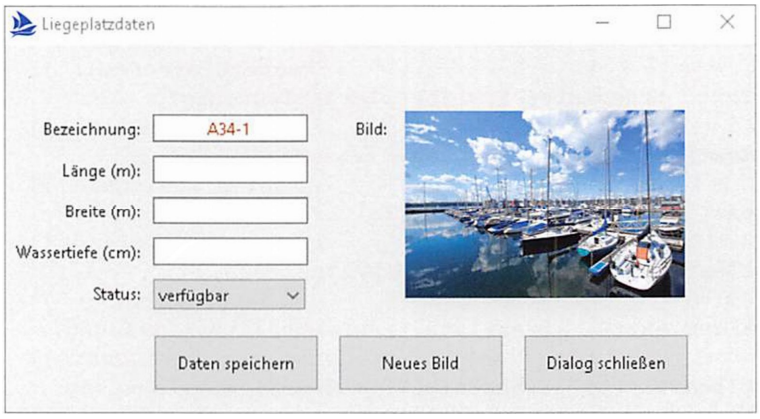
class MainForm(QtWidgets.QWidget,Ui_frmLiegeplatzerfasssung):
    def __init__(self):
        super(MainForm,self).__init__()
        self.setupUi(self)

    def btnSave_click(self):
        pass
    def btnNew_click(self):
        pass
    def btnClose_click(self):
        pass

if __name__ == '__main__':
    app = QApplication(sys.argv)
    form = MainForm()
    form.show()
    sys.exit(app.exec())
```

Hier werden nun die Event-Methoden definiert. Auf die Implementation wird an dieser Stelle verzichtet.

Wird dieser Quellcode ausgeführt, dann erscheint folgendes Fenster für die Erfassung der Daten eines Liegeplatzes

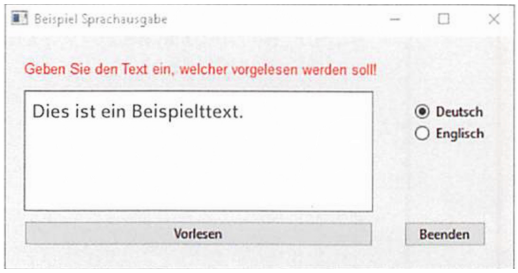


4.2.3. Sprachausgabe

Die Sprachausgabe von Text ist ein Bestandteil, um eine Software barrierefrei zu gestalten und neue Interaktionsformen mit dem Computer zu ermöglichen. Es stehen verschiedene API für die Sprachausgabe zur Verfügung (S. 48). Um diese nutzen zu können, müssen für die jeweilige Programmiersprache entsprechende Bibliotheken zur Verfügung stehen. Eine API, welche mit Python genutzt werden kann, ist die „gTTS-API" von Google (Google Text to Speech API). Die gTTS-API unterstützt mehrere Sprachen, darunter Englisch, Deutsch und Französisch. Um mit dieser API in Python arbeiten zu können, muss sie erst einmal auf dem System installiert werden. Dieses erfolgt unter Windows in der Konsole mit dem Befehl „pip install gTTS".

Weiterhin ist ein Modul zur Soundausgabe notwendig. Hier besteht eine Möglichkeit in der Nutzung von „playsound". Playsound ist ein einfaches Modul, welches plattformübergreifend eingesetzt werden kann und sowohl mp3- als auch wav-Dateien abspielt. Um Playsound zu nutzen, muss das Modul zuerst mit „pip install playsound" installiert werden.

Die Funktionsweise der Sprachausgabe wird an einem Beispiel verdeutlicht. Dazu wird zunächst mit dem QT- Designer die nebenstehende Oberfläche erstellt. Für die Eingabe des Textes wird dieses Mal anstatt eines QLineEdit ein QTextEdit verwendet. Die Sprache soll mittels Radiobutton ausgewählt werden. Exemplarisch wurden hier nur Deutsch und Englisch aufgeführt. Diese Funktionalität kann später beliebig erweitert werden.



Quelltext, welcher vom QT-Designer erstellt wurde.

```
# Form implementation generated from reading ui file ,frm_voice.ui'
#
# Created by: PyQt6 UI code generator 6.2.2
#
# WARNING: Any manual changes made to this file will be lost when pyuic6

# is run again.  Do not edit this file unless you know what you are doing.

from PyQt6 import QtCore, QtGui, QtWidgets
```

```
class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(475, 218)
        self.btnClose = QtWidgets.QPushButton(Form)
        self.btnClose.setGeometry(QtCore.QRect(370, 170, 75, 23))
        self.btnClose.setObjectName("btnClose")
        self.lblText = QtWidgets.QLabel(Form)
        self.lblText.setGeometry(QtCore.QRect(20, 10, 341, 41))
        self.lblText.setStyleSheet("color: rgb(255, 85, 127);\n"
                                   "font: 75 10pt \"MS Shell Dlg 2\";")
        self.lblText.setObjectName("lblText")
        self.btnRead = QtWidgets.QPushButton(Form)
        self.btnRead.setGeometry(QtCore.QRect(20, 170, 321, 23))
        self.btnRead.setObjectName("btnRead")
        self.rbGerman = QtWidgets.QRadioButton(Form)
        self.rbGerman.setGeometry(QtCore.QRect(380, 60, 82, 17))
        self.rbGerman.setChecked(True)
        self.rbGerman.setObjectName("rbGerman")
        self.rbEnglish = QtWidgets.QRadioButton(Form)
        self.rbEnglish.setGeometry(QtCore.QRect(380, 80, 82, 17))
        self.rbEnglish.setObjectName("rbEnglish")
        self.txtText = QtWidgets.QTextEdit(Form)
        self.txtText.setGeometry(QtCore.QRect(20, 50, 321, 111))
        self.txtText.setObjectName("txtText")

        self.retranslateUi(Form)
        self.btnClose.clicked.connect(self.btnClose_click)
        self.btnRead.clicked.connect(self.btnRead_click)
        QtCore.QMetaObject.connectSlotsByName(Form)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Beispiel Sprachausgabe"))
        self.btnClose.setText(_translate("Form", "Beenden"))
        self.lblText.setText(_translate("Form",
                             "Geben Sie den Text ein, welcher vorgelesen
                                                    werden soll!"))
        self.btnRead.setText(_translate("Form", "Vorlesen"))
        self.rbGerman.setText(_translate("Form", "Deutsch"))
        self.rbEnglish.setText(_translate("Form", "Englisch"))
```

Nun ist noch das Hauptprogramm zu programmieren und die Methoden „btnClose_click“ und „btnRead_ click“ sind zu implementieren. Das Verfahren der Sprachausgabe ist unter der Methode „btnRead_click“ zu finden.

Quellcode des Hauptprogramms

```
from frm_voice import Ui_Form
from PyQt6 import QtWidgets
from PyQt6.QtWidgets import QApplication
from gtts import gTTS
from playsound import playsound
import sys, os
```

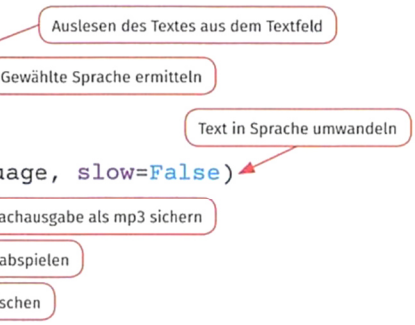
Einbinden der Module für das TTS-System und die Soundausgabe

```
class MainForm(QtWidgets.QWidget,Ui_Form):
    def __init__(self):
        super(MainForm,self).__init__()
        self.setupUi(self)

    def btnRead_click(self):
        output = self.txtText.toPlainText()
        language = 'de'
        if self.rbEnglish.isChecked():
            language = 'en'
        voice_obj = gTTS(text=output, lang=language, slow=False)
        voice_obj.save("output.mp3")
        playsound("output.mp3")
        os.remove("output.mp3")

    def btnClose_click(self):
        app.quit()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    form = MainForm()
    form.show()
    sys.exit(app.exec())
```



Wenn nun, wie im Beispiel angezeigt wurde, ein Text eingegeben und der Button „Vorlesen" gedrückt wird, dann wird der Text in eine mp3-Datei umgewandelt und diese wird dann abgespielt.

❖ Zur Soundausgabe steht der Quelltext im freien Download zur Verfügung und kann ausprobiert werden.

✓ Kompetenzcheck

1. Geben Sie an, welche Aussagen richtig sind.
   a. Tkinter ist die einzige Möglichkeit, wie unter Python grafische Benutzeroberflächen erstellt werden können.
   b. Der QT-Designer ist Bestandteil der Standardbibliothek in Python.
   c. QLineEdit und QTextEdit können für die Texteingabe verwendet werden.
   d. Mit dem QT-Designer können keine Events erstellt werden.
   e. Mit der Methode show() wird in PyQt ein Fenster angezeigt.
   f. Mit der Methode isChecked() kann der Status eines RadioButtons in PyQt abgefragt werden.
   g. Die Farben einzelner GUI-Elemente können in PyQt über die Methode SetStyleSheet(...) gesetzt werden.
2. Installieren Sie den PyQT, gTTS und playsound auf Ihrem Rechner.
3. Binden Sie den QT-Designer und pyuic als Werkzeuge in PyChar ein.
4. Implementieren Sie mithilfe des QT-Designers das GUI-Beispiel von S. 56, experimentieren Sie damit und probieren weitere Dinge aus. Tauschen Sie sich hierbei auch im Klassenverband aus.
5. Implementieren Sie das Beispiel für die Sprachausgabe.
6. Erweitern Sie das Programm zur Sprachausgabe um andere Sprachen, z. B. Französisch.
7. Implementieren Sie einfache und komplexe Benutzerschnittstellen mithilfe von Python indem Sie die Aufgaben 2, 4 und 5 des Kapitels 4 im Handout bearbeiten. Überprüfen Sie außerdem Ihr Wissen zu PyOt mithilfe der Aufgabe 3.

4.3. Benutzerschnittstellen in Java realisieren

➢ **Sie sollen auf der Grundlage von Wireframes und Mockups grafische Benutzerschnittstellen (GUI) in Java realisieren.**

Die Programmierung einer GUI in Java ist sehr komplex und vielfältig (siehe auch Jahrgangsband 2, Modul 8). Deswegen werden auch in diesem Kapitel nur die grundlegendsten GUI-Elemente angesprochen. Durch das Entwickeln von verschiedenen Oberflächen mit unterschiedlichen Anforderungen vertieft und erweitert sich mit der Zeit das Wissen über die einzelnen GUI-Elemente und die GUI-Programmierung in Java im Allgemeinen.



4.3.1. GUI-Programmierung

In Java stehen für die Entwicklung einer grafischen Benutzeroberfläche drei grundlegende API zur Verfügung: AWT, Swing und JavaFX.

Java-API zur Entwicklung grafischer Benutzeroberflächen

AWT
Das AWT (Abstract Window Toolkit) ist das erste Framework von Java. Es stellt verschiedene GUI- Komponenten wie Button, TextField$_f$ Checkbox, List zur Verfügung, zeichnet diese aber nicht selbst, sondern ruft dazu Betriebssystem-Subroutinen auf. Deshalb ist AWT plattformabhängig. Es wurde aber schnell klar, dass das Leistungsangebot von AWT keineswegs für die Zukunft reicht. Heutzutage kommt AWT kaum noch zu Anwendung.

Swing
Swing wurde entwickelt, um bessere GUI-Komponenten als das frühere AWT bereitzustellen. Swing bietet ein Erscheinungsbild, welche das Erscheinungsbild mehrerer Plattformen emuliert. Es hat leistungsfähigere und flexiblere Komponenten als AWT und stellt zu den bekannten Komponenten noch erweiterte Komponenten bereit. Im Gegensatz zu AWT-Komponenten werden Swing-Komponenten nicht durch plattformspezifischen Code implementiert. Stattdessen sind sie vollständig in Java geschrieben und daher plattformunabhängig. Swing ist weiterhin Teil der Standardbibliothek.

JavaFX
JavaFX ist seit Java 11 ein separat gepflegtes Open-Source-Projekt. Es ist in vieler Hinsicht der Nachfolger von Swing und wird aktiv weiterentwickelt. Bei JavaFX sind die API, die Entwickler zum Erstellen von Benutzeroberflächen verwenden, und die Rendering-Engines zum Rendern der Benutzeroberfläche getrennt. Für Windows-Plattformen benutzt die JavaFX-Rendering-Pipeline „DirectSD" und für andere Plattformen die Rendering-Pipelines, Welche auf „OpenGL" basieren. Die Trennung hat den Vorteil, dass die JavaFX API nicht verändert werden müssen, sobald neue Hardware-Rendering-Techniken zur Verfügung stehen. So erlaubt es auch JavaFX, plattformübergreifende Anwendungen zu erstellen, die selbst dann noch laufen, wenn die unterliegenden Implementierungen verändert werden. JavaFX kann die Softwareplattform für das Schreiben umfangreicher Internetanwendungen mit einer modernen Benutzeroberfläche und einem raffinierten Erscheinungsbild bereitstellen. Es ist noch in der Entwicklung und wird voraussichtlich in Zukunft wachsen.

In der Praxis werden sowohl Swing als auch JavaFX verwendet, weshalb man mit beiden APIs vertraut sein sollte. Swing wird größtenteils bei älteren Projekten zu finden sein, welche gepflegt und erweitert werden. Bei Neuentwicklungen wird mit hoher Wahrscheinlichkeit JavaFX eingesetzt, wobei auch hier noch Swing eine Rolle spielt. Allerdings bietet Swing keinen Support für mobile Endgeräte, hier muss ausschließlich auf JavaFX gesetzt werden.

Swing und AWT sollen in naher Zukunft weiter in der Standardbibliothek von Java SE enthalten sein. Aus Sicht der Java-Entwickler sind beide Technologien beim Entwickeln von grafischen Oberflächen nützlich. Mit seiner umfangreichen UI-Komponentenbibliothek kann Swing dem Entwickler einen zusätzlichen Vorteil verschaffen, während JavaFX Swing ersetzen kann, wenn es darum geht, moderne Internetanwendungen oder Anwendungen für den mobilen Bereich zu entwerfen.

Auf den Bereich der Entwicklung von Java-Swing-Oberflächen mithilfe des GUI-Builders wurde bereits im Jahrgangsband 2 eingegangen (siehe Jahrgangsband 2, Modul 8). Deswegen wird nachfolgend die Entwicklung von grafischen Oberflächen mithilfe von JavaFX näher betrachtet. Dazu muss JavaFX auf dem System installiert werden und es soll auch hier ein GUI-Editor, ähnlich dem GUI-Builder, zum Einsatz kommen, damit nicht jedes einzelne GUI-Element mit den entsprechenden Events per Hand implementiert werden muss. Um damit komfortabel arbeiten zu können, muss zunächst Eclipse dementsprechend vorbereitet werden.

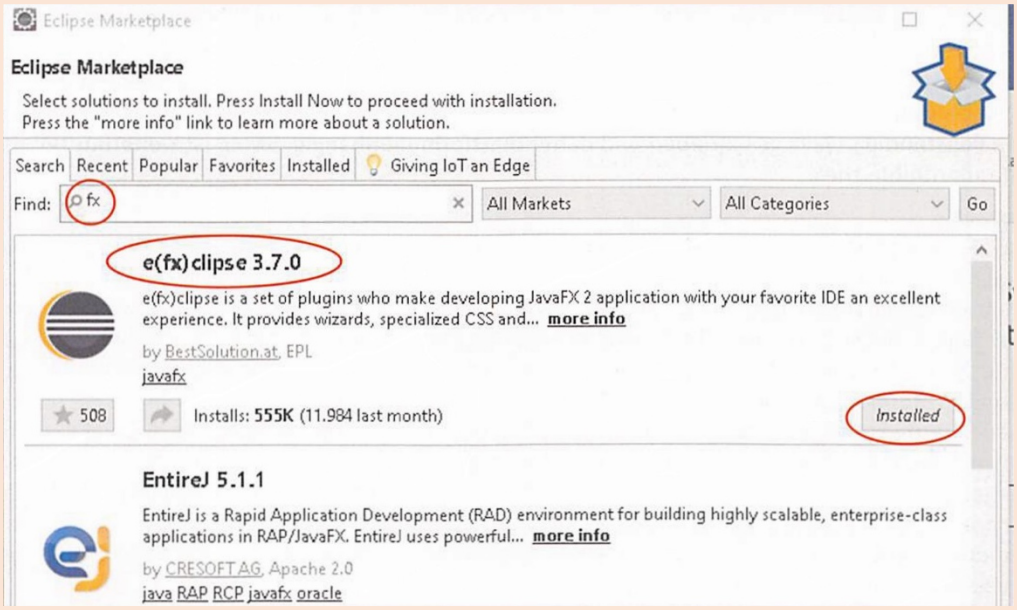
Installieren von JavaFX, Scene-Builder, E(fx)clipse und Einrichten von Eclipse

1 Installieren von JavaFX und dem Scene-Builder
JavaFX kann unter

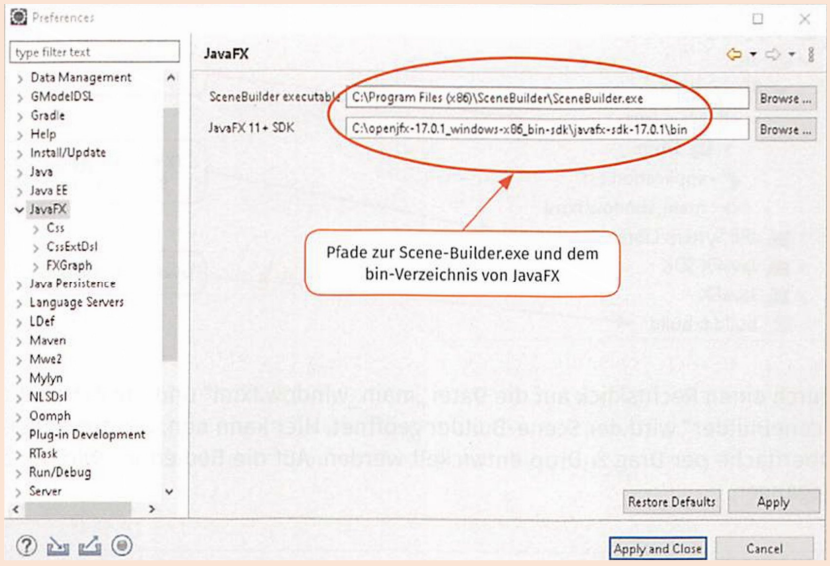
https://gluonhq.com/products/javafx/

heruntergeladen werden. Danach wird JavaFX in einen beliebigen Ordner auf dem Computer entpackt. Der Scene-Builder ist ein GUI-Editor für JavaFX Oberflächen. Er kann unter https://gluonhq.com/products/scene-builder/ heruntergeladen werden und wird danach installiert.

2 Installieren von E(fx)clipse
Zunächst muss e(fx)clipse installiert werden. Dazu wird unter „Help - Eclipse Marketplace ..." das folgende Fenster geöffnet, fx gesucht und das Programm installiert. Danach erfolgt automatisch ein Restart von Eclipse.

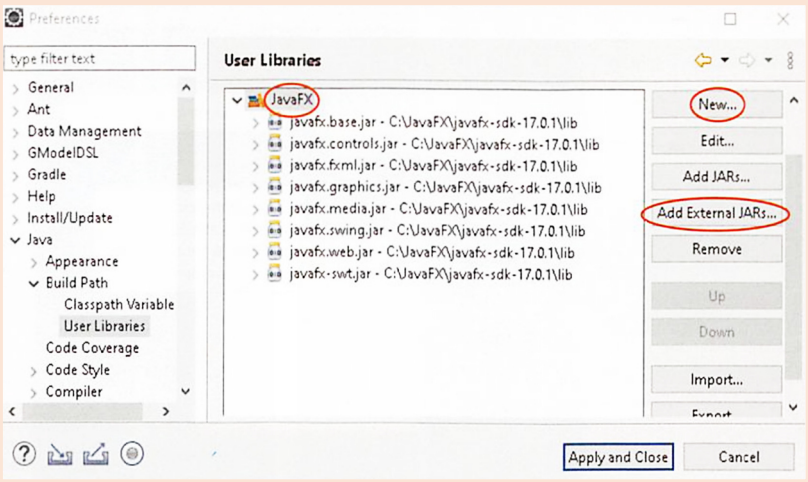


3. Den Path zu JavaFX und den Scene-Builder eintragen
Dazu werden der nachfolgende Dialog unter „Window - Preferences – JavaFX" geöffnet und die entsprechenden Pfade eingetragen.

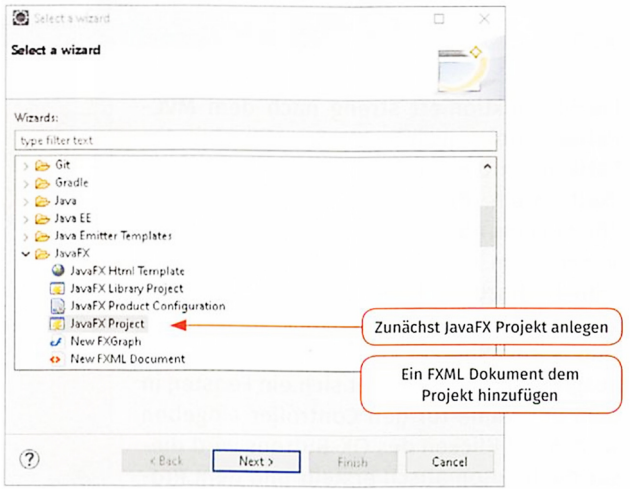


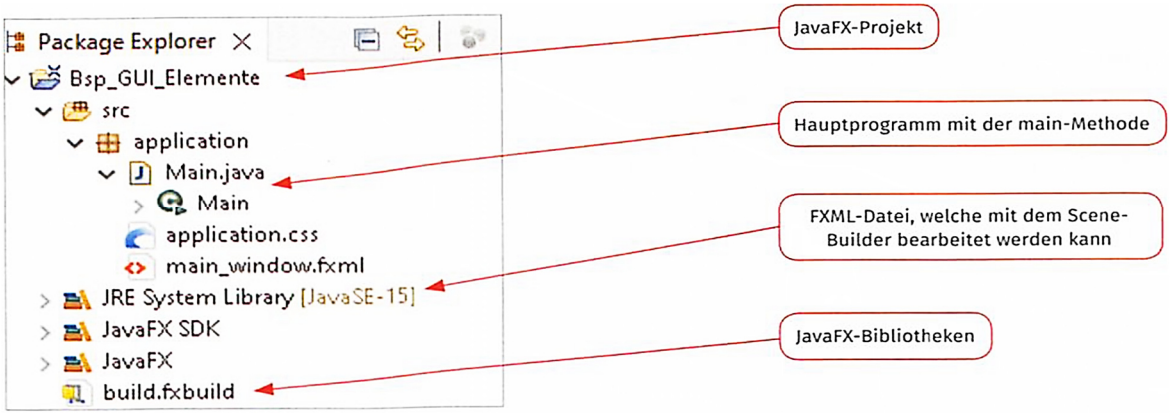
4 Anlegen einer JavaFX-Library

Dazu wird der nachfolgende Dialog unter „Window - Preferences - Java - Build Path - User Libraries" geöffnet. Hier wird unter der „New" ein neuer Eintrag vorgenommen, in diesem Fall JavaFX. Im Anschluss werden JavaFX durch „Add External JARs..." die JARs aus dem Bin-Verzeichnis von JavaFX hinzugefügt.

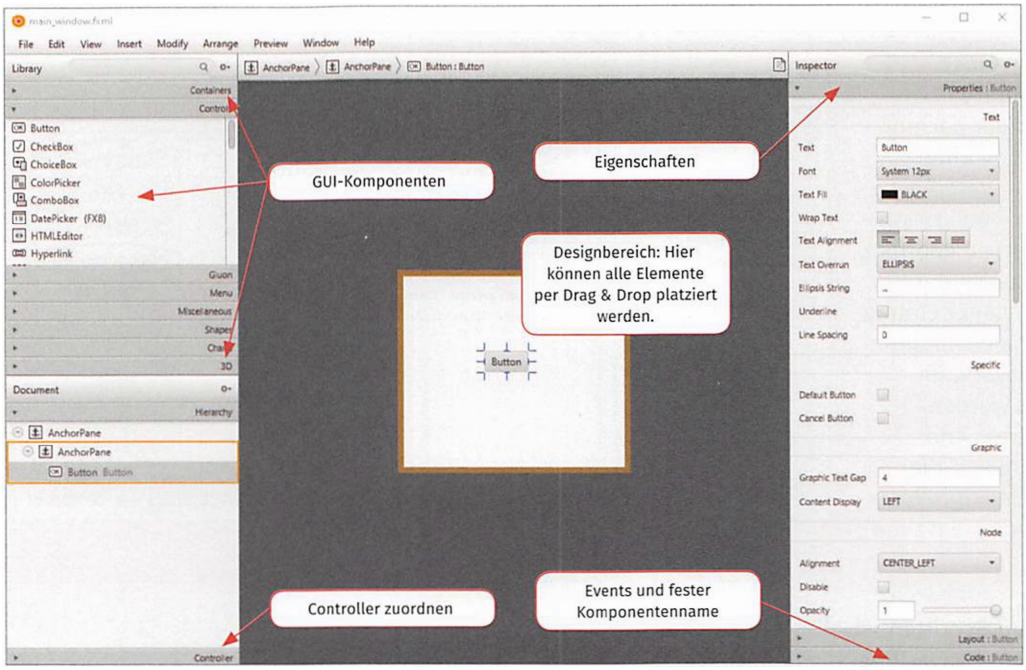


Nach diesen Schritten ist alles eingerichtet und es können Java-Programme mit JavaFX-Oberflachen entwickelt werden. Dazu wird der nebenstehende Dialog mit „File - New - Others" geöffnet und ein JavaFX-Projekt ausgewählt und angelegt. Danach wird dem Projekt durch „Rechtsklick - New - Others" der gleiche Dialog geöffnet und ein FXML-Dokument hinzugefügt. Die erzeugte Datei „module-info.java" wird aus dem Projekt gelöscht, weil diese erst einmal nicht benötigt wird. Außerdem werden dem Projekt noch über ,,Rechtsklick - Build Path - Configure Build Path ..." die JavaFX-Bibliotheken hinzugefügt. Danach sollte der Projektordner wie folgt aussehen:

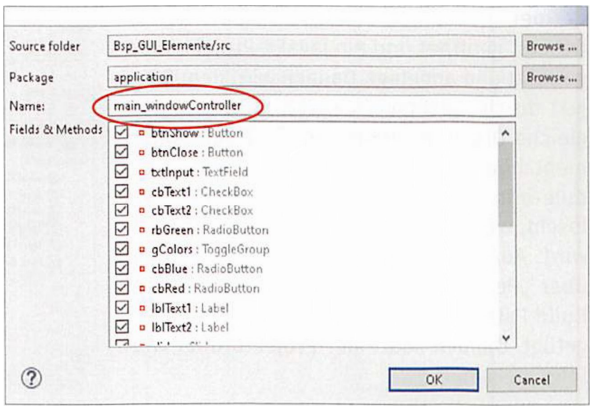




Durch einen Rechtsklick auf die Datei „main_window.fxml" und die Auswahl des Meniipunktes „Open with Scene-Builder" wird der Scene-Builder geöffnet. Hier kann nun, wie aus anderen GUI-Editoren bekannt, die Oberfläche per Drag & Drop entwickelt werden. Auf die Bedienung wird an dieser Stelle nicht näher eingegangen.



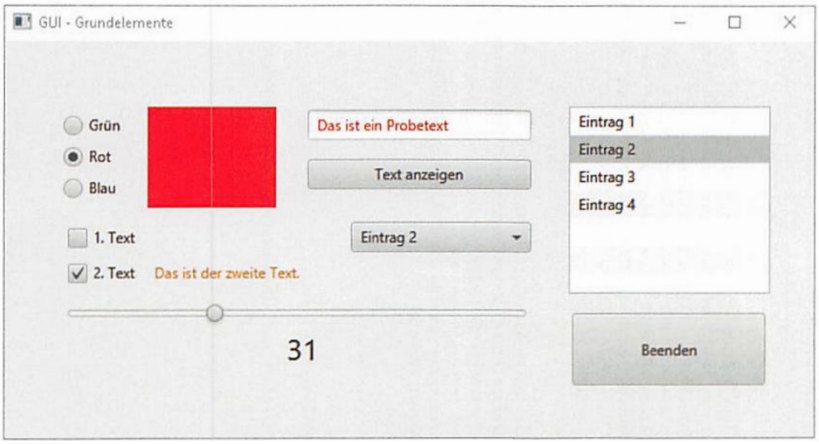
JavaFX funktioniert streng nach dem MVC- Pattern (siehe auch S. 27). Der View-Teil des Patterns wird mit dem Scene-Builder entwickelt. Nach dem Sichern des Entwurfes (View) muss nun noch der Controller erstellt werden, um die Events implementieren zu können. Dazu wird die fxml-Datei im Editor geöffnet und ein Rechtsklick ausgeführt. Nun wird „Source - Generate Controller" ausgewählt und es öffnet sich ein Fenster, in dem der Name für den Controller eingeben wird. Nach Klicken des OK-Buttons wird dieser dann automatisch erstellt und dem Projekt hinzugefügt.



Der Quellcode vom Controller kann nun im Editor geöffnet und die Event-Methoden mit Funktionalität unterlegt werden. Danach wird das Programm auf dem bekannten Weg unter Eclipse ausgeführt.

Nachfolgend wird ein Beispiel mit verschiedenen GUI-Elementen vorgestellt. Dazu wird zunächst ein neues Projekt angelegt und eine fxml-Datei hinzugefügt. Diese wird nun mit dem Scene-Builder bearbeitet.

Die untenstehende Oberfläche wurde mit dem Scene-Builder entwickelt. In diesem Beispiel sind grundlegende GUI-Elemente dargestellt und es wurde eine einfache Funktionalität hinterlegt. Damit soll das Prinzip der Erstellung von grafischen Oberflächen mit dem Scene-Builder und des Event-Handlings dargestellt werden.



Nachfolgend ist der Quellcode aufgeführt und kommentiert, welcher vom Scene-Builder erstellt wurde. Dieser kann als Grundlage genommen werden, für den Fall, dass man doch Elemente per Hand erstellen oder ändern möchte. Normalerweise vermeidet man aber das manuelle Ändern dieses Codes und überlässt die Erstellung komplett dem Scene-Builder.

```xml
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.CheckBox?>
<?import javafx.scene.control.ComboBox?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.ListView?>
<?import javafx.scene.control.RadioButton?>
<?import javafx.scene.control.Slider?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.ToggleGroup?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>

<AnchorPane xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1" fx:Controller="application.main_windowController">
   <children>
      <AnchorPane layoutX="-35.0" layoutY="-28.0" prefHeight="357.0" prefWidth="711.0">
         <children>
            <Button fx:id="btnShow" layoutX="288.0" layoutY="126.0" mnemonicParsing="false" onAction="#btnShow_Click" prefHeight="25.0" prefWidth="186.0" text="Text anzeigen" />
            <Button fx:id="btnClose" layoutX="508.0" layoutY="253.0" mnemonicParsing="false" onAction="#btnClose_Click" prefHeight="60.0" prefWidth="160.0" text="Beenden" />
            <TextField fx:id="txtInput" layoutX="288.0" layoutY="85.0" prefHeight="25.0" prefWidth="186.0" style="-fx-text-fill: red;" />
            <CheckBox fx:id="cbText1" layoutX="89.0" layoutY="182.0" mnemonicParsing="false" onAction="#cbText1_Click" selected="true" text="1. Text" />
            <CheckBox fx:id="cbText2" layoutX="89.0" layoutY="211.0" mnemonicParsing="false" onAction="#cbText2_Click" selected="true" text="2. Text" />
            <RadioButton fx:id="rbGreen" layoutX="85.0" layoutY="89.0" mnemonicParsing="false" onAction="#rbGreen_Click" selected="true" text="Grün" />
               <toggleGroup>
                  <ToggleGroup fx:id="gColors" />
               </toggleGroup>
            </RadioButton>
            <RadioButton fx:id="cbBlue" layoutX="85.0" layoutY="141.0" mnemonicParsing="false" onAction="#cbBlue_Click" text="Blau" toggleGroup="$gColors" />
            <RadioButton fx:id="cbRed" layoutX="85.0" layoutY="115.0" mnemonicParsing="false" onAction="#cbRed_Click" text="Rot" toggleGroup="$gColors" />
            <Label fx:id="lblText1" layoutX="161.0" layoutY="182.0" text="Das ist der erste Text." textFill="#0bb2e0" />
            <Label fx:id="lblText2" layoutX="161.0" layoutY="211.0" text="Das ist der zweite Text." textFill="#de0a0b" />
            <Slider fx:id="slider" layoutX="85.0" layoutY="246.0" max="99.0" onMouseDragged="#slider_moved" prefHeight="14.0" prefWidth="389.0" />
            <Label fx:id="lblValue" layoutX="271.0" layoutY="266.0" prefHeight="35.0" prefWidth="32.0" text="0" />
               <font>
                  <Font size="24.0" />
               </font>
            </Label>
            <ListView fx:id="listView" layoutX="505.0" layoutY="82.0" prefHeight="155.0" prefWidth="167.0" />
            <ComboBox fx:id="comboBox" layoutX="324.0" layoutY="178.0" prefWidth="150.0" />
            <Label fx:id="lblColor" layoutX="155.0" layoutY="82.0" prefHeight="84.0" prefWidth="107.0" style="-fx-background-color: #00E000;" />
         </children>
      </AnchorPane>
   </children>
</AnchorPane>
```

Auf der Grundlage dieses Codes wird nun der Controller erzeugt und die Event-Methoden implementiert. Nachfolgend ist der Quellcode für den Controller aufgeführt. Hier sind die Event-Methoden bereits implementiert.

Quellcode des Controllers

```
package application;

import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.control.ToggleGroup;
import java.net.URL;
import java.util.ResourceBundle;
import javafx.application.Platform;
import javafx.event.ActionEvent;
import javafx.scene.control.Label;
import javafx.scene.control.Slider;
import javafx.scene.control.ListView;
import javafx.scene.control.ComboBox;
import javafx.scene.control.RadioButton;
import javafx.scene.control.CheckBox;
import javafx.scene.input.DragEvent;

public class main_windowController implements Initializable {
        @FXML
        private Button btnShow;
        @FXML
        private Button btnClose;
        @FXML
        private TextField txtInput;
        @FXML
        private CheckBox cbText1;
        @FXML
        private CheckBox cbText2;
        @FXML
        private RadioButton rbGreen;
        @FXML
        private ToggleGroup gColors;
        @FXML
        private RadioButton cbBlue;
        @FXML
        private RadioButton cbRed;
        @FXML
        private Label lblText1;
        @FXML
        private Label lblText2;
        @FXML
        private Slider slider;
        @FXML
        private Label lblValue;
        @FXML
        private ListView listView;
        @FXML
        private ComboBox comboBox;
        @FXML
        private Label lblColor;

        // Event Listener on Button[#btnShow].onAction
        @FXML
```

Import aller benötigten Bibliotheken

Definition der GUI-Elemente

```
    public void btnShow_Click(ActionEvent event) {
        Platform.runLater(new Runnable() {
            public void run() {
                Alert alert = new Alert(Alert.AlertType.
                                                    INFORMATION);
                alert.setTitle("Information");
                alert.setHeaderText(txtInput.getText());
                alert.showAndWait();
            }
        });
    }
    // Event Listener on Button[#btnClose].onAction
    @FXML
    public void btnClose_Click(ActionEvent event) {
        System.exit(0);
    }
    // Event Listener on CheckBox[#cbText1].onAction
    @FXML
    public void cbText1_click(ActionEvent event) {
        if (cbText1.isSelected()) lblText1.setVisible(true);
        else lblText1.setVisible(false);
    }
    // Event Listener on CheckBox[#cbText2].onAction
    @FXML
    public void cbText2_Click(ActionEvent event) {
        if (cbText2.isSelected()) lblText2.setVisible(true);
        else lblText2.setVisible(false);
    }
    // Event Listener on RadioButton[#rbGreen].onAction
    @FXML
    public void rbGreen_Click(ActionEvent event) {
        lblColor.setStyle("-fx-background-color: #00E000");
    }
    // Event Listener on RadioButton[#cbBlue].onAction
    @FXML
    public void cbBlue_Click(ActionEvent event) {
        lblColor.setStyle("-fx-background-color: blue");
    }
    // Event Listener on RadioButton[#cbRed].onAction
    @FXML
    public void cbRed_Click(ActionEvent event) {
        lblColor.setStyle("-fx-background-color: red");
    }
    // Event Listener on Slider[#slider].onMouseDragged
    @FXML
    public void slider_moved() {
        lblValue.setText(String.valueOf((int)slider.getValue()));
    }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        comboBox.getItems().addAll("Eintrag 1",
                                   "Eintrag 2",
                                   "Eintrag 3",
                                   "Eintrag 4");
        comboBox.setValue("Eintrag 1");
        listView.getItems().addAll("Eintrag 1",
                                   "Eintrag 2",
                                   "Eintrag 3",
                                   "Eintrag 4");
    }
}
```

Anzeigen einer Messagebox

Der Combobox und der ListView Einträge hinzufügen

An dem erzeugten Quellcode des Hauptprogramms müssen noch einige Veränderungen vorgenommen werden. Zunächst muss die Zeile

```
BorderPane root = new BorderPane();
```

durch

```
Parent root = FXMLLoader.load(getClass().getResource("main_window.fxml"));
```

ersetzt werden. In der Zeile

```
Scene scene = new Scene(root,400,400);
```

werden die Größenangaben „400,400" entfernt und die Zeile

```
scene.getStylesheets().add(getClass().getResource("application.css").
                                                    toExternalForm());
```

wird komplett gelöscht.

Nachfolgend ist der erneuerte Quellcode des Hauptprogramms aufgeführt.

Quellcode des Hauptprogramms

```
package application;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.stage.Stage;
import javafx.scene.Parent;
import javafx.scene.Scene;

public class Main extends Application {
        @Override
        public void start(Stage primaryStage) {
                try {
                        Parent root = FXMLLoader.load(getClass().
                                                getResource("main_window.fxml"));
                        Scene scene = new Scene(root);
                        primaryStage.setTitle("GUI - Grundelemente");
                        primaryStage.setScene(scene);
                        primaryStage.show();
                } catch(Exception e) {
                        e.printStackTrace();
                }
        }

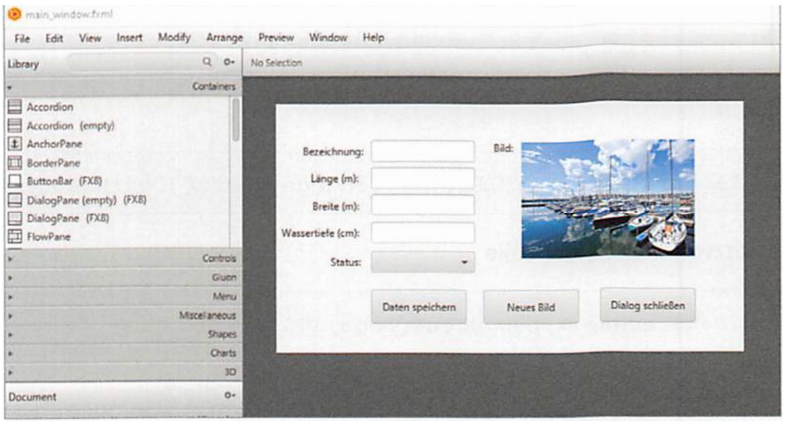
        public static void main(String[] args) {
                launch(args);
        }
}
```

Diese Zeile ist neu.

Größenangaben wurden entfernt

4.3.2. Umsetzung der Beispielaufgabe

Die Oberfläche für die Beispielaufgabe wurde wieder unter Zuhilfenahme des Scene-Builders entwickelt. Nachdem alle GUI-Elemente auf der Oberfläche angeordnet sind und die entsprechenden Eigenschaften gesetzt wurden, können nun die Events, z. B. Buttonklicks definiert und implementiert werden.



Um die Events festzulegen, wird ebenfalls der Scene-Builder verwendet. Nachdem alles gespeichert wurde, wird im Anschluss der Controller generiert und im Scene-Builder zugewiesen. Der Quellcode des Controllers kann nun geöffnet und die Events können implementiert werden. In diesem Beispiel wurde auf Implementierung der Funktionalität verzichtet, weil es hier vor allem um die Erstellung der Benutzeroberfläche geht. Das Thema der Implementierung der Funktionalität wird im Kapitel 2 thematisiert. Der Quellcode der fxml-Datei, des Controllers und des Hauptprogramms sind nachstehend aufgeführt.

```
<?xml version="1.0" encording="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ComboBox?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.image.AnchorPane?>

<AnchorPane prefHeight="281.0" prefWidth="407.0" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1" fx:Controller="application.main_windowController">
   <children>
      <AnchorPane layoutX="-266.0" layoutY="-206.0" prefHeight="200.0" prefWidth="200.0" />
      <AnchorPane layoutX="-35.0" layoutY="-28.0" prefHeight="357.0" prefWidth="711.0">
         <children>
            <Button fx:id="btnSave" layoutX="93.0" layoutY="216.0" mnemonicParsing="false" onAction="#btnSave_click" prefHeight="39.0" prefWidth="111.0" text="Daten speichern" />
            <Button fx:id="btnNew" layoutX="224.0" layoutY="216.0" mnemonicParsing="false" onAction="#btnNew_click" prefHeight="39.0" prefWidth="111.0" text="Neues Bild" />
            <Button fx:id="btnClose" layoutX="360.0" layoutY="216.0" mnemonicParsing="false" onAction="#btnClose_click" prefHeight="39.0" prefWidth="111.0" text="Dialog schließen" />

            <TextField fx:id="txtBezeichnung" alignment="CENTER" layoutX="93.0" layoutY="42.0" prefHeight="25.0" prefWidth="121.0" style="-fx-text-fill: brown;" />
            <TextField fx:id="txtLaenge" layoutX="93.0" layoutY="73.0" prefHeight="25.0" prefWidth="121.0" />
            <TextField fx:id="txtBreite" layoutX="93.0" layoutY="104.0" prefHeight="25.0" prefWidth="121.0" />
            <TextField fx:id="txtWassertiefe" layoutX="93.0" layoutY="137.0" prefHeight="25.0" prefWidth="121.0" />

            <Label fx:id="lblBezeichnung" layoutX="161.0" layoutY="182.0" text="Das ist der erste Text." textFill="#0bb2e0" />
            <Label fx:id="lblLaenge" layoutX="161.0" layoutY="182.0" text="Das ist der erste Text." textFill="#0bb2e0" />
            <Label fx:id="lblBreite" layoutX="161.0" layoutY="182.0" text="Das ist der erste Text." textFill="#0bb2e0" />
            <Label fx:id="lblWassertiefe" layoutX="161.0" layoutY="182.0" text="Das ist der erste Text." textFill="#0bb2e0" />
            <Label fx:id="lblStatus" layoutX="161.0" layoutY="182.0" text="Das ist der erste Text." textFill="#0bb2e0" />
            <Label fx:id="lblBild" layoutX="161.0" layoutY="182.0" text="Das ist der erste Text." textFill="#0bb2e0" />

            <ImageView fx:id="imgView" fitHeight="200.0" fitWidth="200.0" layoutX="271.0" layoutY="51.0" pickOnBounds="true" preserveRatio="#true" />
               <image>
                  <Image backgroundLoading="true" url="@Liegeplatz.jpg" />
               </image>
            </ImageView>
            </ComboBox fx:id="cbStatus" layoutX="93.0" layoutY="171.0" prefHeight="25.0" prefWidth="121.0" />
         </children>
      </AnchorPane>
   </children>
</AnchorPane>
```

Quellcode des Controllers

```java
package application;

import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.scene.control.Label;
import javafx.scene.control.ComboBox;
import javafx.scene.image.ImageView;

public class main_windowController implements Initializable {
	@FXML
	private Button btnSave;
	@FXML
	private Button btnNew;
	@FXML
	private Button btnClose;
	@FXML
	private TextField txtBezeichnung;
	@FXML
	private TextField txtLaenge;
	@FXML
	private TextField txtBreite;
	@FXML
	private TextField txtWasserteife;
	@FXML
	private Label lblBezeichnung;
	@FXML
	private Label lblLaenge;
	@FXML
	private Label lblBreite;
	@FXML
	private Label lblWassertiefe;
	@FXML
	private Label lblStatus;
	@FXML
	private Label lblBild;
	@FXML
	private ImageView imgView;
	@FXML
	private ComboBox cbStatus;

	// Event Listener on Button[#btnSave].onAction
	@FXML
	public void btnSave_click(ActionEvent event) {
		// TODO Autogenerated
	}
	// Event Listener on Button[#btnNew].onAction
	@FXML
	public void btnNew_click(ActionEvent event) {
		// TODO Autogenerated
	}
```

```
        // Event Listener on Button[#btnClose].onAction
        @FXML
        public void btnClose_click(ActionEvent event) {
                // TODO Autogenerated
        }

        @Override
        public void initialize(URL url, ResourceBundle rb) {
                cbStatus.getItems().addAll("verfügbar",
                                           "vermietet",
                                           "belegt",
                                           "gesperrt");
                cbStatus.setValue("verfügbar");
        }
}
```

Quellcode des Hauptprogramms

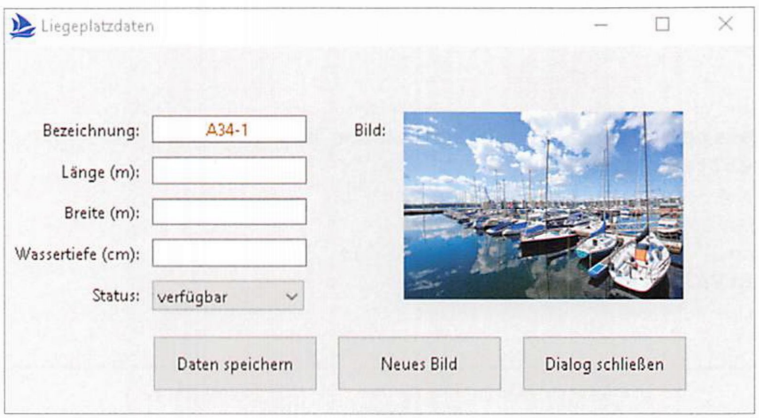
```
package application;

import javafx.scene.image.Image;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.stage.Stage;
import javafx.scene.Parent;
import javafx.scene.Scene;

public class Main extends Application {
        @Override
        public void start(Stage primaryStage) {
                try {
                        Parent root = FXMLLoader.load(getClass().
                                                getResource("main_window.fxml"));
                        Scene scene = new Scene(root);
                        primaryStage.setTitle("Liegeplatzdaten");
                        primaryStage.getIcons().add(
                        new Image(this.getClass()
                                        .getResource("icon.png").toString()));
                        primaryStage.setScene(scene);
                        primaryStage.show();
                } catch(Exception e) {
                        e.printStackTrace();
                }
        }

        public static void main(String[] args) {
                launch(args);
        }
}
```

Wird das Programm ausgeführt, dann erscheint folgendes Fenster für die Erfassung der Daten eines Liegeplatzes.

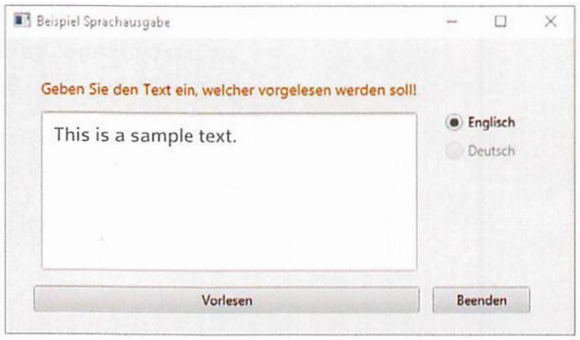


4.3.3. Sprachausgabe

Die Sprachausgabe von Text ist ein Bestandteil, um eine Software barrierefrei zu gestalten und neue Interaktionsformen mit dem Computer zu ermöglichen. Es stehen verschiedene API für die Sprachausgabe in Java zur Verfügung (siehe S. 48). So kann beispielsweise die Cloud-Version von Google genutzt werden. Dazu muss aber ein entsprechendes Konto eingerichtet werden. An dieser Stelle soll eine einfache Variante der Sprachausgabe unter Verwendung von FreeTTS vorgestellt werden.

- FreeTTS ist eine Implementierung der Java Speech API und steht frei zur Verfügung, kommt aber in der Soundqualität nicht an die kommerzielle Version von Google heran. Um FreeTTS zu nutzen, muss es zunächst heruntergeladen und die Bibliotheken müssen in Eclipse eingebunden werden. Offizielle Seite und Download von FreeTTS: https://freetts.sourceforge.io/

Für ein Beispielprogramm wird zunächst mit dem SceneBuilder die nebenstehende Oberfläche erstellt. Für die Eingabe des Textes wird dieses Mal anstatt eines TextFileds ein TextArea verwendet. Die Sprache soll mittels Radiobutton ausgewählt werden. Da standardmäßig in FreeTTS nur Englisch zur Verfügung steht, wurde der Radiobutton für „Deutsch" zunächst „disabled" Später können andere „Voices" ergänzt werden. Nachdem die Oberfläche erstellt wurde, wird nun wieder ein Controller erzeugt und hier werden die Events für die ,,Button-Clicks implementiert. Nachfolgend ist der Quelltext für den Controller aufgeführt.



Quelltext für den Controller

```
package application;
import javafx.fxml.FXML;
import javafx.scene.control.Button;

import com.sun.speech.freetts.Voice;
import com.sun.speech.freetts.VoiceManager;

import javafx.event.ActionEvent;
import javafx.scene.control.RadioButton;
import javafx.scene.control.TextArea;
import javafx.scene.control.ToggleGroup;

public class main_windowController {

        @FXML
        private Button bthClose;
        @FXML
        private Button btnRead;
        @FXML
        private RadioButton rbGerman;
        @FXML
        private ToggleGroup gLanguages;
        @FXML
        private RadioButton rbEnglish;
        @FXML
        private TextArea txtText;

        @FXML
        public void btnClose_click(ActionEvent event) {
               System.exit(0);
        }
        @FXML
        public void btnRead_click(ActionEvent event)  {
              System.setProperty("freetts.voices",
                           "com.sun.speech.freetts.en.us.cmu_us_kal.
                                                      KevinVoiceDirectory");
              Voice voice = VoiceManager.getInstance().getVoice("kevin");
              if (voice != null) {
                 voice.allocate();
              }
              try {
                 voice.setRate(160);
                 voice.setPitch(150);
                 voice.setVolume(3);
                 String text = txtText.getText();
                 voice.speak(text);
              }
              catch(Exception e) {
                 e.printStackTrace();
              }
        }
}
```

Import der Bibliotheken für die Sprachausgabe

Standardstimme auswählen

Setzen von Parametern für die Stimme.

Vorlesen des Textes

Quellcode des Hauptprogramms

```
package application;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.stage.Stage;
import javafx.scene.Parent;
import javafx.scene.Scene;

public class Main extends Application {
   @Override
   public void start(Stage primaryStage) {
     try {
       Parent root = FXMLLoader.load(getClass().getResource("main_window.
                                                              fxml"));
       Scene scene = new Scene(root);
       primaryStage.setTitle("Beispiel Sprachausgabe");
       primaryStage.setScene(scene);
       primaryStage.show();
     } catch(Exception e) {
       e.printStackTrace();
     }
   }

   public static void main(String[] args) {
       launch(args);
   }
}
```

Wenn nun, wie im Beispiel angezeigt, ein Text eingegeben und der Button „Vorlesen" gedrückt wird, dann erfolgt die entsprechende Sprachausgabe.

❖ Zur Soundausgabe steht der Quelltext im freien Download zur Verfügung und kann ausprobiert werden.

✓ Kompetenzcheck

1. Geben Sie an, welche Aussagen richtig sind.

   a. JavaFX stellt die einzige Möglichkeit dar, grafische Oberflächen in Java zu erstellen.
   b. Mit dem Scene-Builder können Java-Swing-Oberflächen entwickelt werden.
   c. JavaFX verwendet das MVC-Entwurfsmuster.
   d. Aus der fxml-Datei, welche mit dem Scene-Builder bearbeitet wurde, kann ein Controller generiert werden.
   e. Im Controller wird die Ereignisbehandlung implementiert.
   f. Mit der Methode „isChecked()" der entsprechenden Checkbox kann ermittelt werden, ob die Checkbox ausgewählt ist oder nicht.

2. Installieren Sie JavaFX und den Scene-Builder auf Ihrem Rechner.

3. Richten Sie Eclipse ein, indem Sie E(fx)clipse installieren, den Pfad zu JavaFX und dem Scene-Builder eintragen und eine JavaFX-Bibliothek anlegen.

4. Implementieren Sie mithilfe des Scene-Builders das GUI-Beispiel von S. 70, experimentieren damit und probieren Sie weitere Dinge aus. Tauschen Sie sich hierbei auch im Klassenverband aus.

5. Implementieren Sie das Beispiel für die Sprachausgabe.

6. Erweitern Sie das Beispiel der Sprachausgabe um eine weitere Sprache.

7. Implementieren Sie einfache und komplexe Benutzerschnittstellen mithilfe von Java, indem Sie die Aufgaben 2, 4 und 5 des Kapitels 4 im Handout bearbeiten. Überprüfen Sie außerdem Ihr Wissen zu JavaFX mithilfe der Aufgabe 3.

4.4. Webbasierte Benutzerschnittstellen realisieren

➢ **Sie sollen webbasierte Benutzerschnittstellen mithilfe von HTML, CSS und JavaScript entwickeln.**



Webauftritte sind für Unternehmen unverzichtbar, denn ohne eine eigene Webseite wird ein Unternehmen fast nicht mehr wahrgenommen. Gesucht wird heutzutage nach Dienstleistungen und Angeboten insbesondere mithilfe von Smartphones, Tablets und PCs im Internet. Dazu bedarf es webbasierter Benutzerschnittstellen. Um solche zu entwickeln, kommen verschiedene Sprachen und Technologien zum Einsatz. Die grundlegendste Sprache dabei ist HTML. Mithilfe von HTML wird der Inhalt und das grundlegende Layout einer Webseite beschrieben. Neben HTML werden i.d.R. CSS verwendet, um das Erscheinungsbild einer Webseite zu beschreiben, und die Sprache JavaScript, um die Funktionalität einer Webseite zu implementieren.

Zum Ansehen einer solchen Webseite ist zudem ein Webbrowser nötig. Wenn Nutzer eine Webseite ansehen möchten, dann ruft der Webbrowser den erforderlichen Inhalt von einem Webserver ab und zeigt die Seite auf dem Gerät der Nutzer an. Bekannte Browser sind beispielsweise Google Chrome, Mozilla Firefox, Microsoft Edge, Apple Safari oder Opera.

Wegen der hohen Bedeutung der Internetpräsenz wird auch im Rahmen des Projekts „Yachthafen Resort" ein Webauftritt von JIKU IT-Solutions entwickelt. Im Zusammenhang damit soll exemplarisch nachfolgendes Beispiel umgesetzt werden.



4.4.1. HTML

HTML (Hypertext Markup Language) ist eine Auszeichnungssprache, welche zur Strukturierung und Gestaltung von elektronischen Dokumenten dient. Vor allem findet HTML bei der Erstellung von Internetseiten Anwendung. Hier bildet HTML die Basis und wird durch Scriptsprachen wie JavaScript oder PHP erweitert.

Geschichtliche Entwicklungvon HTML

HTML 1.0
wurde 1993 veröffentlicht. Damals wurden nicht viele Webseiten erstellt und die Möglichkeiten von HTML waren sehr eingeschränkt.

HTML 2.0
wurde 1995 veröffentlicht, enthielt die komplette Funktionalität von HTML 1.0 und fügte noch einige neue Funktionen hinzu. HTML 2.0 war bis Januar 1997 Standard für die Webentwicklung und definierte erstmals viele HTML-Kernfunktionen.

HTML 3.0
enthielt viele neue und verbesserte Funktionen für HTML und bot weitaus leistungsfähigere Funktionen. Leider

unterstützten viele Browserentwickler diesen Standard nur unzureichend und so wurde die HTML-3.0-Spezifikation aufgegeben. Mit der Zeit wurde es immer offensichtlicher, dass ein Standard gefunden werden musste. Zu diesem Zweck wurde 1994 das „World Wide Web Consortium" (W3C) gegründet und dieses entwickelte HTML 3.2, welches 1997 zum offiziellen Standard wurde.

Dieser Standard wurde vom W3C schnell zu HTML 4.0 weiterentwickelt und 1998 veröffentlicht. Nachdem HTML 4.0 erschienen war, wurde es in einigen kleineren Schritten zu **HTML 4.01** korrigiert, welches dann die endgültige Version war. Um den neuen Anforderungen der Webentwicklung gerecht zu werden, wurden viele Ideen, z. B. XHTML, entwickelt. Allerdings entschied das W3C, dass HTML weiterhin die Zukunft des Webs sein soll und andere Ideen wurden eingestellt.

So wurde HTML 5.0 2014 zum neuen Standard der Webentwicklung, welcher zumindest für das nächste Jahrzehnt gelten soll. Für HTML 5.0 wurden die meisten Teile aus HTML 4.1 übernommen und um viele neue Elemente, Attribute und Fähigkeiten ergänzt. Die aktuelle Version ist HTML 5.2, welche im Dezember 2017 veröffentlicht wurde.

HTML ist keine Programmiersprache im herkömmlichen Sinne, da Kontroll- und Ablaufstrukturen sowie Variablen fehlen. In HTML wird der Inhalt eines Dokuments in elementare Strukturelemente, sogenannte Tags, aufgesplittet. Jeder Tag kennzeichnet den Anfang seines Gültigkeitsbereiches. Die Tags werden in spitze Klammern eingeschlossen, z. B. <html>. Zu den Tags können zusätzlich ein oder mehrere Attribute angegeben werden. Die Attribute enthalten zusätzliche Informationen für den einzelnen Tag. Dazu wird das entsprechende Attribut angegeben und diesem ein Wert zugewiesen, z. B.:

```
<img src="beispiel.jpg" width="500" height="600">
```

Jeder Tag enthält i.d.R. auch ein Endkennzeichen. Als Endkennzeichen wird der Tag wiederholt. Es wird jedoch ein „/" vorangestellt, z. B. </html>. Die einzelnen Tags können ineinander verschachtelt werden. Auf diese Weise entsteht eine hierarchische Struktur.

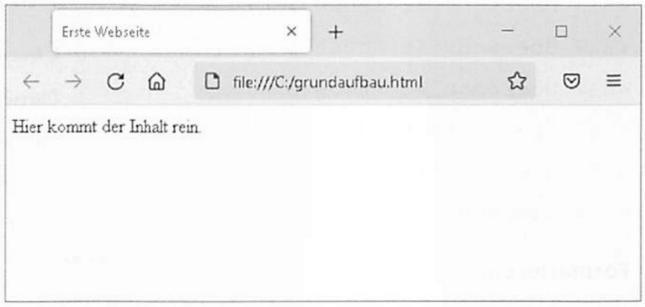
Ein HTML5-Dokument, welches dem vom W3C vorgegebenen Standard entspricht, enthält mindestens vier Tags, welche zusammen den Grundaufbau für das HTML-Dokument bilden. Diese sind „html", „head", „title" und „body". In der Regel kommt dann noch eine Doctyp-Information hinzu, welche über die Art des HTML-Dokuments Auskunft gibt.

Grundaufbau einer HTML-Seite

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Erste Webseite </title>
  </head>
  <body>
        Hier kommt der Inhalt rein.
  </body>
</html>
```

- html: startet und beendet das HTML-Dokument
- head: enthält Informationen über den im „body“ notierten Text
- title: wird nicht auf der Webseite angezeigt, aber z. B. im Tab des Browsers
- body: alles, was hier eingefügt wird, wird auf der Webseite angezeigt

Um eine HTML-Seite zu erzeugen, muss nun dieser Quelltext eingegeben und in einer Datei mit der Endung „html“ abgespeichert werden. Diese kann dann anschließend mithilfe eines Browsers aufgerufen und angezeigt werden. Zur Eingabe des Quellcodes wird ein Editor als Texterfassungsprogramm benötigt. Dieses kann ein einfacher Texteditor oder ein spezieller HTML-Editor sein. In diesem Fall wurde Visual Studio Code verwendet, welches ein plattformübergreifender, Sehr verbreiteter Texteditor ist. Das Ergebnis ist nebenstehend abgebildet.



❖ Nicht alle Browser unterstützen den gleichen HTML-Standard, sodass gleiche Webseiten auf verschiedenen Browsern zu unterschiedlichen Darstellungen führen können.

Neben dem eben erläuterten Verfahren können Webseiten auch mithilfe von WYSIWYG-Editoren (What You See Is What You Get) entwickelt werden. Hier sind zunächst nicht einmal HTML-Kenntnisse notwendig, da der HTML-Code von der Anwendung selbst generiert wird. Es wird praktisch nur die HTML-Seite designt. Später können auch hier Änderungen direkt am Quellcode vorgenommen werden. Eine weitere Vorgehensweise in der Praxis ist es, normale Dokumente, z. B. Word-Dokumente, in HTML zu konvertieren. Dazu gibt es zahlreiche Tools. Auch kann die integrierte Funktionalität des entsprechenden Textverarbeitungsprogramms genutzt werden.

Übersicht über einige Werkzeuge zur Erstellung von Webseiten	
Texteditoren	Visual Studio Code, Phase 5, Atom, Sublime Text 3, Android Studio, Brackets
WYSIWYG-Editoren	Dreamweaver, Google Web Designer, Microsoft Frontpage, Microsoft Expression Web, RapidWeaver
Textverarbeitungssoftware mit HTML-Unterstützung	LibreOffice Writer, Microsoft Word, TextEdit von Apple

Trotzdem sind Grundfähigkeiten zum Lesen und Erstellen von HTML-Dokumenten sinnvoll, um eventuelle Fehler oder unerwünschte Formate zu erkennen und zu beseitigen. Zur Erklärung wird im weiteren Verlauf auf wesentliche HTML-Sprachelemente eingegangen.

❖ Eine ausführliche Dokumentation von HTML, CSS und JavaScript ist u.a. auf der Seite von SELFHTML https://wiki.selfhtml.org/ zu finden. Diese Dokumentation steht auch offline zur Verfügung.

Wie zuvor erwähnt wird ein Text in HTML durch sogenannte Tags strukturiert. Je nach verwendetem Tag werden verschiedene Strukturen erzeugt. Fast alle Strukturen werden durch ein einleitendes und ein abschließendes Tag markiert. Nachfolgend sind einige wichtige Tags aufgeführt.

Wichtige HTML-Tags

Überschriften

```
<h1> Überschrift Ebene 1 </h1>
<h2> Überschrift Ebene 2 </h2>
<h3> Überschrift Ebene 3 </h3>
<h4> Überschrift Ebene 4 </h4>
<h5> Überschrift Ebene 5 </h5>
<h6> Überschrift Ebene 6 </h6>
```

Kennzeichnung von Überschriften

Überschrift Ebene 1

Überschrift Ebene 2

Überschrift Ebene 3

Überschrift Ebene 4

Überschrift Ebene 5

Überschrift Ebene 6

Formatierung

`<p> … </p>`	Kennzeichnung von Absätzen
` `	Zeilenumbruch
`<hr>`	horizontale Linie

Textauszeichnungen

`<b> Text fett</b>`	b (bold) – Text wird fett dargestellt
`<i> Text kursiv </i>`	i (italic, kursiv) – Text wird kursiv dargestellt

Tabelle

```
<table>
  <tr>
    <th> ArtikelNr </th>
    <th> Bezeichnung </th>
    <th> Preis </th>
  </tr>
  <tr>
    <td> 12 </td>
    <td> Postkarte </td>
    <td> 1.30 </td>
  </tr>
  <tr>
    <td> 13 </td>
    <td> Zeitung </td>
    <td> 2.50 </td>
  </tr>
</table>
```

<table> – Tabelle
<tr> – Tabellenzeile
<th> – Spaltenüberschrift
<td> – Tabellenfeld

ArtikelNr	Bezeichnung	Preis
12	Postkarte	1.30
13	Zeitung	2.50

Listen	
`<ul>` `  <li> Unterhaltung</li>` `  <li> Sauna</li>` `  <li> Wellness</li>` `</ul>`	• Unterhaltung • Sauna • Wellness – unnummerierte Liste – Listeneintrag
`<ol>` `  <li> Müller, Frank </li>` `  <li> Schulz, Caroline</li>` `  <li> Maier, Stefanie</li>` `</ol>`	1. Müller, Frank 2. Schulz, Caroline 3. Maier, Stefanie – nummerierte Liste – Listeneintrag
Verweise, Links	
`<a href="https://www.google.de/">` `  Suche mit Google</a>`	Suche mit Google href – links auf die entsprechende Seite
Bilder, Audio, Video	
`<img src="sonnenaufgang_01.jpg"` `alt="Sonnenaufgang am Meer">`	src – Bilddatei alt – Alternative Bildbeschreibung für Barrierefreiheit 
`<picture>` `  <source media="min-width:800px"` `  srcset="sonnenaufgang_01.jpg">` `</picture>`	Ab HTML5; hiermit können für verschiedene Endgeräte Grafiken in verschiedenen Auflösungen vorgegeben werden. Für die unterschiedlichen Grafiken wird das Tag „source“ verwendet. media – Informationen über das Endgerät abfragen srcset – Bilddatei
`<audio controls autoplay>` `  <source src="meeresrauschen.mp3"` `  type="audio/mpeg">` `  Ihr Browser unterstützt nicht` `  die Wiedergabe der Audiodatei.` `</audio>`	Ab HTML5; spielt Audiodateien in HTML ab; mit dem Tag „source“ können wieder unterschiedliche Audiodateien angegeben werden. src – Audiodatei type – Typ der Audiodatei
`<video controls>` `  <source src="sonnenaufgang.mp4"` `  type="video/mp4">` `  Die Wiedergabe des Videos wird von` `  Ihrem Browser nicht unterstützt.` `</video>`	Ab HTML5; spielt Videodateien in HTML ab; mit dem Tag „source“ können wieder unterschiedliche Videodateien angegeben werden. src – Videodatei type – Typ der Videodatei

Sonderzeichen

Um sicherzustellen, dass der Browser die korrekte Zeichencodierung nutzt, muss diese im <head>-Bereich angeben werden (siehe auch S. 84, Anweisung <meta charset=„utf-8“>). So können auch Sonderzeichen wie Emoticons oder ganz andere Schriftsätze verwendet werden.

Eine weitere Möglichkeit besteht darin, die in HTML definierten Sonderzeichen zu verwenden. Diese beginnen mit einem und werden mit „;“ abgeschlossen.

Auswahl von Sonderzeichen und deren Darstellung im HTML-Code					
Zeichen	**HTML-Darstellung**	**Zeichen**	**HTML-Darstellung**	**Zeichen**	**HTML-Darstellung**
ä	ä	ö	ö	ü	ü
Ä	Ä	Ö	Ö	Ü	Ü
ß	ß	§	§	&	&
¼	¼	½	½	¾	¾
<	<	>	>	Leerzeichen	

4.4.2. CSS

Mithilfe der aufgeführten Tags von HTML kann der Inhalt einer Webseite strukturiert werden und diese ist auch bereits benutzbar. Allerdings sieht eine so gestaltete Seite in der Regel wenig ansprechend aus. An diesem Punkt kommt nun CCS (Cascading Stylesheets) zum Einsatz. Hiermit wird das Design einer Webseite gestaltet. So kann z. B. mithilfe von CSS die Farbe und die Größe von Texten festgelegt werden. Die aktuelle Version ist CSS3. Es gibt mehrere Möglichkeiten, um CSS in das HTML-Dokument zu integrieren. In der Regel wird die zweite oder die dritte Variante verwendet, weil man das Design vom Inhalt trennen möchte. Das ist auch die ursprüngliche Idee hinter CSS. Durch die letztgenannte Variante ist es sogar möglich, das Design von Webseiten durch das Austauschen der CSS-Datei schnell und ohne großen Aufwand zu ändern.

Möglichkeiten der Einbindung von CSS

Direkt dem HTML-Tag als style-Attribut hinzufügen

```
<h1 style="text-align: center;
color:green; "font-weight: bold">
Überschrift </h1>
<p style="color: blue;"> Blauer Text. </p>
<p style="text-align: right; color: brown;">
Brauner Text. </p>
```



Im head-Bereich als style-Element hinzufügen

```
<!doctype html>
<html lang="de">
  <head>
    <style>
    h1 {
        text-align: center;
        background-color: darkcyan;
        color: whitesmoke;
    }
    p {
        text-align: center;
        font-size: 26px;
        color: orange;
    }
    </style>
 <title> Titel</title>
  </head>
  <body>
      <h1>Überschrift</h1>
      <p>Das ist ein Beispieltext</p>
  </body>
</html>
```

Gilt für alle h1-Überschriften im Dokument

Gilt für alle Absätze im Dokument



Erstellen einer externen CSS-Datei und als Link im head-Bereich hinzufügen

```
<html lang="de">
  <head>
    <link rel="stylesheet" href="stylesheet.css">
    …
```

Diese Vorgehensweise wird immer dann angewandt, wenn die Style-Definition im head-Bereich zu umfangreich wird oder mehrere Webseiten das gleiche Design erhalten sollen.

Falls man verschieden designte Abschnitte haben möchte, können auch sogenannte Klassen im head- Bereich definiert und diese den einzelnen Abschnitten zugewiesen werden. Dazu wird nach einem einleitenden Punkt ein Name für eine Klasse gewählt. In geschweiften Klammern werden die Eigenschaften definiert, welche für diese Klasse gelten sollen. Im HTML-Code wird diese Klasse dann den entsprechenden Tags zugeordnet.

Klassen im head-Bereich definieren

```
<!doctype html>
<html lang="de">
  <head>
    <style>
    .absatz_design1 {          ← Definition der ersten Klasse
        text-align: left;
        font-size: 18px;
        color: green;
    }
    .absatz_design2 {          ← Definition der zweiten Klasse
        text-align: center;
        font-size: 26px;
        color: orange;
    }
    </style>
 <title> Titel</title>
  </head>
  <body>
     <p class="absatz_design1"> Das ist der erste Beispieltext mit
                                                    Design1 </p>
     <p class="absatz_design2"> Das ist der zweite Beispieltext mit
                                                    Design2 </p>
     <p class="absatz_design1"> Das ist der dritte Beispieltext wieder
                                                  mit Design1 </p>

  </body>
</html>
```

Zuweisen der Klassen mithilfe des Attributes „class" zu verschiedenen Abschnitten

Das ist der erste Beispieltext mit Design1

Das ist der zweite Beispieltext mit Design 2

Das ist der dritte Beispieltext wieder mit Design1

Einer Klasse (Selektor) oder einem „style" können die unterschiedlichsten Eigenschaften mit entsprechenden Werten zugeordnet werden. Nachfolgend werden einige Eigenschaften und mögliche Werte exemplarisch aufgelistet.

Eigenschaft	Beschreibung	Mögliche Werte
background-color	Hintergrundfarbe	transparent, Farbewert, Farbenname
background-image	Hintergrundbild	url(), none
border-color	Rahmenfarbe	transparent, Farbewert, Farbenname
border-style	Rahmenart	none, dotted, dashed, solid, double, groove, inset, outset
border-width	Rahmenbreite	thin, medium, thick, absolute Zahl
color	Farbe	transparent, Farbewert, Farbenname
float	Text umfließend	left, right, none
font-family	Schriftfamilie	Schriftart
font-size	Schriftgröße	absolute Zahl, Prozentsatz, small, medium, large
font-style	Schriftstil	italic, oblique, normal
font-weight	Schriftgewicht	normal, bold
letter-spacing	Zeichenabstand	normal, absolute Zahl
line-height	Zeilenhöhe	normal, Zahl
margin-bottom	Außenabstand unten	absolute Zahl, Prozentsatz, auto
margin-left	Außenabstand links	absolute Zahl, Prozentsatz, auto
margin-right	Außenabstand rechts	absolute Zahl, Prozentsatz, auto
margin-top	Außenabstand oben	absolute Zahl, Prozentsatz, auto
padding	Innenabstand	absolute Zahl, Prozentsatz
padding-bottom	Innenabstand unten	absolute Zahl, Prozentsatz
padding-left	Innenabstand links	absolute Zahl, Prozentsatz
padding-right	Innenabstand rechts	absolute Zahl, Prozentsatz
padding-top	Innenabstand oben	absolute Zahl, Prozentsatz
text-align	Textausrichtung	left, right, center, justify
text-decoration	Textdekoration	none, underline, overline, line-through, blink

Mithilfe von HTML und CSS können nun ansprechende Webseiten entwickelt werden, z. B. die Startseite des Webauftritts des „Yachthafen Resorts", welche im Rahmen des Gesamtprojektes entwickelt wird.

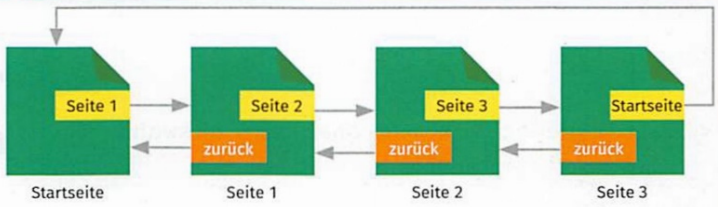
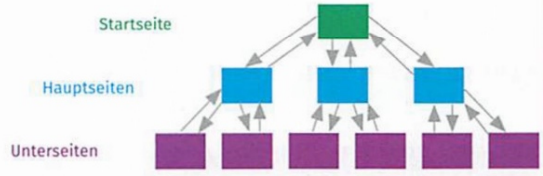
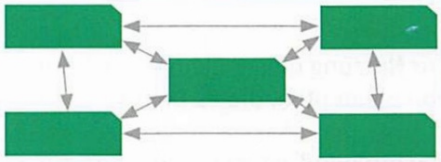


4.4.3. Navigation und Struktur einer Webseite

In der Regel besteht ein Webauftritt nicht nur aus einer Seite, sondern es sind mehrere Seiten durch entsprechende Links miteinander verknüpft. Dabei können die einzelnen Webseiten in Startseite, Haupt- und Unterseiten unterteilt werden.

Arten von Webseiten eines Webauftrittes	
Startseite (Homepage)	Die Startseite eines Webauftrittes ist der Ausgangspunkt für alle weiteren Seiten. Sie ist die erste Seite, die Besucher sehen, wenn sie eine URL laden. Sie enthält u. a. Informationen über die Inhalte des Webauftritts, ansprechende Bilder und es gibt eine Menüleiste, in der die Hauptseiten verlinkt sind.
Hauptseiten	Hauptseiten enthalten Informationen über einen bestimmten größeren Abschnitt oder Bereich des Webauftritts. Zum Beispiel könnten beim Yachthafen Resort die Bereiche Übernachtung, Liegeplätze, Wellness oder Events auf Hauptseiten dargestellt werden.
Unterseiten	Unterseiten enthalten detaillierte Informationen zu Inhalten der Hauptseite, z. B. könnte es eine Unterseite „Anfrage Liegeplatz" geben, welche von der Hauptseite „Liegeplätze" aufgerufen wird.

Diese einzelnen Seiten sind durch Links untereinander verknüpft. Die Art der Verknüpfung der Seiten untereinander bildet die Struktur eines Webauftrittes. Bei der Wahl der Verknüpfung ist besonders wichtig, dass Benutzer alle Unterseiten schnell und einfach finden können (siehe auch UX Design). Dabei werden verschiedene Strukturen unterschieden.

Verknüpfungsstrukturen von Webseiten	
Lineare Struktur	Die einfachste Art Webseiten zu organisieren besteht darin, diese hintereinander anzuordnen. Diese Struktur entspricht der von Büchern oder Zeitschriften. Die Reihenfolge kann chronologisch sein, eine logische Reihe von Themen, die vom Allgemeinen zum Spezifischen übergehen, oder alphabetisch wie in Indizes, Enzyklopädien und Glossaren. Diese Art der Struktur eignet sich beispielsweise für Schulungen oder Umfragen, bei denen sich der Benutzer durch einen festen Satz von Material durcharbeitet. Die einzigen Links auf den Seiten sind diejenigen, die den linearen Navigationspfad ermöglichen. 
Baumstruktur	Die Baumstruktur ist die Struktur, welche am häufigsten vorkommt. Da Websites in der Regel um eine einzelne Startseite herum organisiert sind, die dann auf Haupt- und Unterseiten verweist, eignet sich die Baumstruktur besonders für die Organisation von Webauftritten. Die meisten Webauftritte verwenden eine Form von mehrstufiger Baumarchitektur. Die Unterteilung in Haupt- und Unterkategorien hat einen starken Vorteil für komplexe Webauftritte, da die meisten Benutzer mit einer Baumstruktur vertraut sind. 
Netzstruktur	Bei dieser Struktur sind alle Seiten untereinander direkt verknüpft. Diese Struktur ist für Webauftritte i. d. R. nicht geeignet, weil sie für Benutzer schwer zu verstehen ist und man sich leicht verliert. Diese Struktur funktioniert am besten bei kleinen Webauftritten, wo der Überblick leicht zu behalten ist. 

Heutzutage sind bei den meisten Webauftritten Mischformen der vorgestellten Strukturen zu finden, um Benutzer optimal bei der Benutzung des Webauftrittes zu unterstützen.

4.4.4. Formulare und JavaScript

Mit den bisherigen Mitteln und ein wenig Gespür für grafisches Design ist man bereits in der Lage, ordentliche Webauftritte zu erstellen. Allerdings werden den Benutzern bis jetzt nur Informationen in aufbereiteter Form zur Verfügung gestellt. Die Interaktion mit der Webanwendung besteht für die Nutzer aber hauptsächlich in der Navigation zwischen den einzelnen Seiten. Eine Eingabe und Weiterverarbeitung von Werten in irgendeiner Form sind noch nicht möglich. Aber auch für diesen Bereich gibt es in HTML und CSS entsprechende Elemente, um grundlegende Oberflächenelemente auf der Webseite zu nutzen. In diesem Bereich wird vor allem das input-Element sehr viel genutzt.

Auswahl grundlegender Oberflächenelemente

HTML-Code	Anzeige
`<input type="text" value="Beispieltext">`	Beispieltext
`<input type="button" value="Berechnen">`	Berechnen
`<input type="range" value="40">`	
`<input type="checkbox" checked> Auswahl`	Auswahl
`input type="radio" checked> Auswahl`	Auswahl
`<input type="color" value="#aa00cc">`	
`<input type="month" value="2022-01">`	Januar 2022
`<label for="fname">Bezeichnung:</label>`	Bezeichnung:
`<meter value= "0.4"></meter>`	
`<textarea rows="3" cols="10">` `Das ist ein längerer Text` `</textarea>`	Das ist ein längerer Text

Zur Nutzung dieser Elemente wird zunächst ein Webformular erstellt. Ein Webformular ist eine Sammlung von Elementen, die es Benutzern erlauben, mit der Webanwendung zu interagieren.

Nachfolgend ist der Quellcode für das Webformular zur Liegeplatzanfrage im Rahmen des Webauftritts des „Yachthafen Resorts" aufgeführt. Die Lösung ist als eine erste Variante im Entwicklungszyklus von Benutzerschnittstellen zu betrachten und soll die grundlegenden Elemente eines Webformulars exemplarisch darstellen. Deshalb wurden auch weitere Interaktionselemente wie Eingabefelder für den Namen, für ein Datum usw. weggelassen. Diese wären im Rahmen eines gelungenen UX Design notwendig.

Quellcode für das Webformular zur Liegeplatzanfrage

```
<html lang="de">
  <head>
   <meta charset="utf-8">
   <style>
     body { background-color:lightblue; }
     h1 {
         color: darkblue;
         font-weight: bold; }
     p { margin-bottom:1em; }
     input { margin-bottom:2em; }
     textarea { margin-bottom:2em; }
     label { color: darkblue;
             font-weight: bold; }
   </style>
   <title>Liegeplatzanfrage</title>
  </head>
  <body>
   <h1>Liegeplatzanfrage</h1>
   <p>Sie können uns kontaktieren, indem Sie das
      Formular ausfüllen und an uns absenden. <br>
      Wir nehmen dann mit Ihnen über die angegebenen
      E-Mail-Adresse Kontakt auf.</p>
   <form action="https://..." method="post">
    <label for="email" style="margin-bottom:20em;">
          Ihre E-Mail-Adresse: </label><br>
    <input type="email" name="email" size=35><br>
    <label for="typ">Yachttyp: </label><br>
    <input type="radio" name="typ" value="Segelyacht" checked> Segelyacht
    <input type="radio" name="typ" value="Motoryacht">Motoryacht<br>
    <label for="laenge">Länge der Yacht: </label><br>
    <input type="text" name="laenge" size=10 text-align=right> m<br>
    <label for="breite">Breite der Yacht: </label><br>
    <input type="text" name="breite" size=10 text-align=right> m<br>
    <label for="tiefgang">Tiefgang der Yacht: </label><br>
    <input type="text" name="tiefgang" size=10 text-align=right> cm<br>
    <label for="anfrage">Anfrage: </label><br>
    <textarea name="nachricht" rows="10" cols="50" ></textarea><br>
    <input type="checkbox"> Ich stimme zu, dass meine Daten gespeichert
                                                    werden. <br>
    <input type="submit" value="Absenden">
    <input type="reset" value="Abbrechen">
   </form>
  </body>
</html>
```

Head-Bereich, wo unter anderem auch verschiedene Styles definiert werden

Beginn des body-Bereichs

Beginn des form-Bereichs mit unterschiedlichen Interaktions-elementen

Buttons zum Versenden und zum Abbrechen

Mit den Tags „<form>" und „</ form>" wird ein Formular definiert. Im einleitenden <form>-Tag erscheint zuerst das Pflichtattribut „action". Hier wird angegeben, an welche Ressource die Formulardaten beim Kicken des submit-Buttons zu übertragen sind. Die URL kann hier beispielsweise die Referenz auf ein Programm auf dem Server-Rechner sein, dass die Formulardaten weiterverarbeitet und damit die Aktion ausführt.

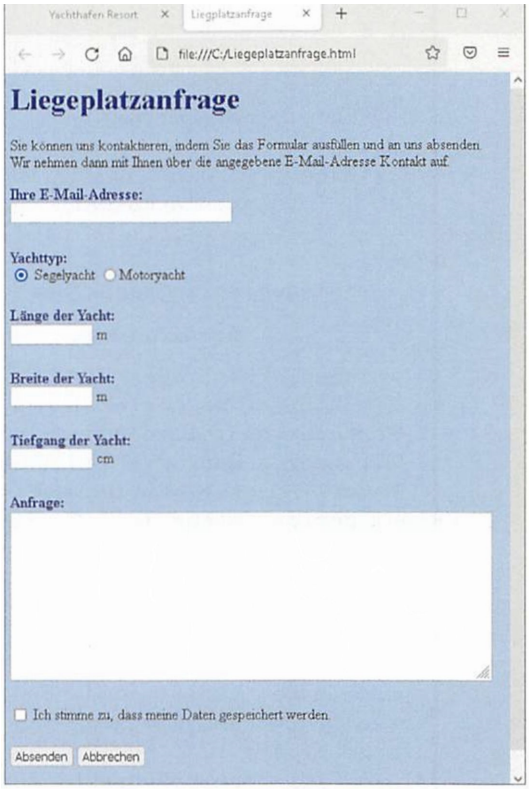
Das zweite Attribut bei der Formulardefinition ist das Attribut „method". Es bestimmt, welche HTTP-Methode (siehe auch Kapitel 2.3.7, S. 189) genutzt werden soll. Dabei gibt es zwei Möglichkeiten:

1. method=„get": Hier hängt der Browser die Formulardaten als Parameter an die Aufrufadresse an. Dieses ist die Standardeinstellung.
2. method="post": Hier übermittelt der Browser die Formulardaten mit einer POST-Anforderung an die Aufrufadresse und die Daten werden dem verarbeitenden Programm über den Standardeingabekanal zur Verfügung gestellt.

Nach der Empfehlung des W3C ist die get-Variante immer dann zu wählen, wenn das auswertende Programm die Daten nur zur Ablaufsteuerung benötigt, z. B. zur Anmeldung oder für eine Suche.

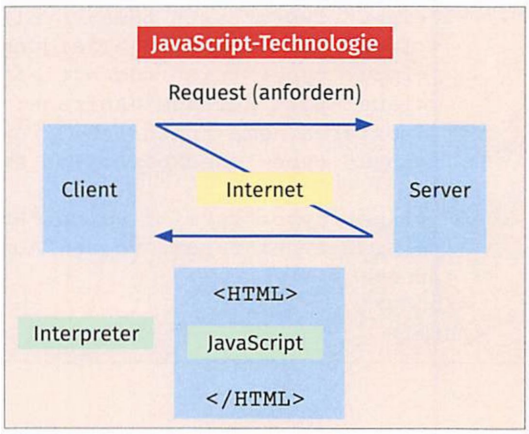
Mit der post-Variante können größere Datenmengen übermittelt werden. Sie wird für Fälle empfohlen, in denen die Daten über das auswertende Programm hinaus weiterverarbeitet werden sollen, z. B. zur Speicherung in einer Datenbank.

Dieses ist auch im vorgestellten Beispiel der Fall, weshalb die post-Variante gewählt wurde. Zwischen den form-Tags können nun die benötigten Interaktionselemente definiert werden. Das Ergebnis, wenn das Webformular in einem Browser aufgerufen wird, ist nebenstehend abgebildet.



JavaScript

Durch die Möglichkeiten im Zusammenhang mit Webformularen werden Benutzern bereits einige Interaktionsmöglichkeiten zur Verfügung gestellt. Leider existiert in HTML keine Möglichkeit, Formularinhalte selbst auszuwerten. Formulare in HTML können nur Inhalte sammeln und müssen diese Daten dann an andere Instanzen zur Auswertung weiterleiten. Einen Ausweg bieten hier kleine Programme, die in den HTML-Code eingefügt werden. Diese sind in einer Programmiersprache, z.B. JavaScript oder PHP, geschrieben und werden dann im Browser ausgeführt. Dabei wird PHP überwiegend für die serverseitige Programmierung verwendet. Im Zusammenhang mit der clientseitigen Programmierung der Benutzerschnittstelle ist JavaScript die mit Abstand am meisten verwendete Programmiersprache. JavaScript ist eine Interpreter-Sprache, welche auch das objektorientierte Programmierparadigma (siehe auch Jahrgangsband 2, Modul 8) unterstützt. Mit JavaScript lassen sich daher sowohl kleine Algorithmen als auch komplexe Framework schreiben. JavaScript ist an die



Programmiersprache „Java" (siehe auch Jahrgangsband 2, Modul 8) angelehnt, aber wesentlich simpler aufgebaut. Wenn JavaScript-Code in ein HTML-Dokument integriert wird, dann kann beispielsweise der Inhalt von Textfeldern ausgelesen, bearbeitet und neu geschrieben werden. Um JavaScript-Code in ein HTML-Dokument einzubinden, gibt es mehrere Möglichkeiten.

Möglichkeiten der Einbindung von JavaScript in HTML-Dokumente

Direktes Einfügen im body-Bereich

Dazu wird das Tag „script“ verwendet. Zwischen <script> und </script> wird dann der JavaScript-Code eingefügt.

```
<!DOCTYPE html>
<html>
   <body>
      <h2>JavaScript im Body</h2>
      <script>
         document.write("Das ist JavaScript im Body");
      </script>
   </body>
</html>
```

JavaScript-Code einbinden

Indirektes Einfügen

Hierbei wird ein Verweis auf eine JavaScript-Datei in den head- oder body-Bereich eingefügt. Diese wird dann aufgerufen und interpretiert.

```
<script src="dateiname.js" type="text/javascript"> </script>
```

Um in JavaScript ein Programm zu schreiben, stehen wie in jeder anderen Programmiersprache verschiedene Sprachelemente und Kontrollstrukturen zur Verfügung. Die Syntax von JavaScript ist sehr stark an der Syntax von Java angelehnt, weshalb es nicht schwerfällt, die Syntax von JavaScript zu verstehen, wenn man mit Java vertraut ist. Aus diesem Grund wird in der nachfolgenden Tabelle nur ein kleiner Überblick über wichtige Sprachelemente von JavaScript gegeben.

Überblick über JavaScript-Syntax	
Kommentare	`// Einzeilenkommentar  /* Mehrzeilenkommentar */`
Rechenoperatoren	`+, -, *, /, % - Modulo, ** - Exponential`
Vergleichsoperatoren	`>, >=, <, <=, ==, !=`
Logische Operatoren	`&& - Und, || - Oder, ! - Nicht`
Anweisung	`let x = 0;`
Verzweigung	`if ( Bedingung) { … }` `else { … }`
Kopfgesteuerte Schleife	`while (Bedingung) {` `  …` `}`
Zählschleife	`for (Startwert; Bedingung; Schrittweite) {` `  …` `}`
Umwandeln von Zeichenketten	`parseInt(string)  parseFloat(string)`

Um mithilfe von JavaScript beispielsweise einzelne Elemente einer HTML-Seite anzusprechen, Text hinzuzufügen oder Events auszuwerten, können verschiedene Methoden verwendet werden. Einige davon sind in der nachstehenden Tabelle aufgeführt.

Im Rahmen des Webauftritts des „Yachthafen Resorts" soll nun ein Formular entwickelt werden, in dem Kunden

Methoden	
createElement()	erzeugt ein neues HTML-Element
createTextNode()	erzeugt einen neuen Text
getElementById()	greift auf ein Element über eine ID zu
querySelector()	gibt das erste Element zurück, das dem angegebenen Selektor entspricht
hasFocus()	ermittelt, ob ein Element den Fokus hat
open()	öffnet einen HTML-Outputstream
close()	schließt einen HTML-Outputstream
write()	schreibt etwas in ein Dokument
writeln()	schreibt etwas in ein Dokument mit Zeilenumbruch
addEventListener()	erzeugt einen Eventhandler
removeEventListener()	entfernt einen Eventhandler
createEvent()	erzeugt ein neues Event

zusätzliche Wellnessleistungen buchen können. Dieses wäre noch ohne JavaScript möglich. Gleichzeitig soll es dem Kunden aber ermöglicht werden, sich die voraussichtlichen Gesamtkosten berechnen und anzeigen zu lassen. Hier kommt nun JavaScript zum Einsatz. Auch für dieses Beispiel gilt, dass es nur grundsätzlich die Verwendung von JavaScript in HTML-Dokumenten verdeutlichen soll.

Quellcode des Formulars zur Buchung von Wellnessleistungen

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        body {background-color:lightblue;}
        h1 {color: darkblue;
            font-weight: bold; }
        p {margin-bottom:1em; }
        input { margin-bottom:2em; }
        label {color: darkblue;
            font-weight: bold; }
    </style>
    <title>Zusaltzleistungen Wellness</title>
    <script type="text/javascript">
        function berechneGesamtBetrag() {
            var gesamt = 0;
            var cbrm = document.getElementById("cbrm");
            var cbmb = document.getElementById("cbmb");
            if (cbrm.checked) {
                var anzahl_rm = document.getElementById("anzahl_rm").
                                                                value;
            if (anzahl_rm >= 0 && anzahl_rm <= 10) {
                gesamt = gesamt + (anzahl_rm *45.00);
            } else {
                alert("Geben Sie bitte einen Wert im Bereich von 0 bis
                                                                10 ein.");
```

Definition des Styles von einzelnen Elementen

Hier beginnt der JavaScript-Code

Definition einer Funktion

Referenzen auf die Checkboxen ermitteln

Wenn die Checkboxen ausgewählt sind, dann die Anzahl auslesen und mit dem Wert der Leistung multiplizierte und dem Gesamtbetrag hinzurechnen

```
                return;
            }
        }
        if (cbmb.checked) {
            var anzahl_mb = document.getElementById("anzahl_mb").
                                                        value;
            if (anzahl_mb >= 0 && anzahl_mb <= 10) {
                gesamt = gesamt + (anzahl_mb *30.00);
            } else {
                alert("Geben Sie bitte einen Wert im Bereich von 0 bis
                                                        10 ein.");
                return;
            }
        }
        document.getElementById("gesamt").value = gesamt;
    }
  </script>
</head>
<body>
    <h1> Zusatzleistungen im Wellnessbereich</h1>
    <form action="..." method="post">
      <input type="checkbox" id="cbrm"> Rückenmassage 30 min (Preis:
                                                        45,00 Euro)
      <label for="anzahl_rm"style="margin-left:1em"> Anzahl: </label>
      <input type="number" id="anzahl_rm" size=2 value=0 min=0 max=10
             oninput="this.value=this.value.replace(/[^0-9]/g,'');">
      <br>
      <input type="checkbox" id="cbmb"> Moorbad 20 min (Preis:
                                                        30,00 Euro)
      <label for="anzahl_mb"style="margin-left:1em"> Anzahl: </label>
      <input type="number" id="anzahl_mb" size=2 value=0 min=0 max=10
             oninput="this.value=this.value.replace(/[^0-9]/g,'');"><br>
      <label for="gesamt"> Gesamtbetrag: </label>
      <input type="text" id="gesamt" size=10 readonly style="text-
                                                        align:right"> Euro
      <input type="button" id="berechne" value="Berechnen"
             style="margin-left:1em" onclick="berechneGesamtBetrag()">
      <br>
      <input type="submit" value="Leistung buchen">
      <input type="reset" value="Abbrechen">
    </form>
  </body>
</html>
```

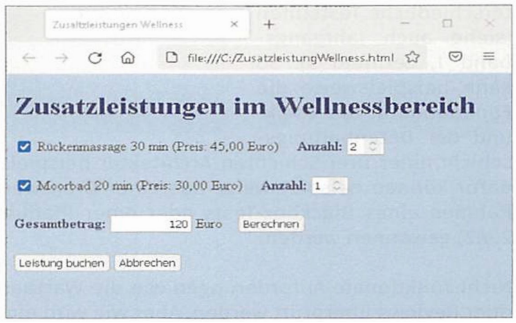
Abfragen, ob die Anzahl im Bereich von 0 bis 10 liegt; wenn nicht, dann eine Meldung ausgeben

Gesamtbetrag anzeigen

Im body-Bereich die Interaktionselemente in einem Formular anlegen

Die im head-Bereich definierte JavaScript-Funktion bei einem Onclick-Ereignis aufrufen

Wenn das HTML-Dokument in einem Browser ausgeführt wird, dann ist nebenstehende Ausgabe zu sehen. Hier können Benutzer nun ihre Zusatzleistungen auswählen, eine gewünschte Anzahl eingeben und sich durch Klicken des „Berechnen"-Buttons den voraussichtlichen Gesamtbetrag berechnen und anzeigen lassen.



Bei der Definition der Eingabefelder für die Anzahl wurde bei der Umsetzung folgender JavaScript-Code miteingefügt:

Dieser JavaScript-Code soll auf einfache Weise sicherstellen, dass nur ganze Zahlen eingegeben werden können.

```
oninput="this.value=this.value.replace(/[^0-9]/g,'');"
```

✓ Kompetenzcheck

1. Geben Sie an, welche Aussagen richtig sind.

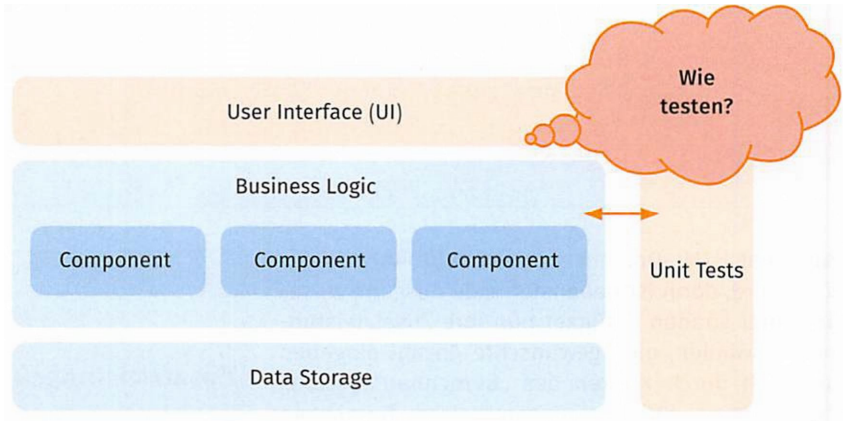
   a. HTML ist eine Programmiersprache.
   b. Durch HTML-Tags wird ein HTML-Dokument strukturiert.
   c. Ein HTML-Dokument hat einen head- und einen body-Bereich.
   d. Mit dem Tag <headline> werden Überschriften in HTML gekennzeichnet.
   e. CSS dient vor allem dazu, das Design vom Inhalt zu trennen.
   f. CSS kann nur im head-Bereich im HTML-Dokument integriert werden.
   g. Mit der CSS-Eigenschaft „color" kann die Hintergrundfarbe gesetzt werden.
   h. Mithilfe von Formularen kann die Interaktion von Benutzer mit der Webseite gestaltet werden.
   i. JavaScript ist ein Bestandteil von Java, welcher hauptsächlich für die Entwicklung von Webseiten dient.
   j. Um JavaScript in HTML einzubinden, muss zunächst immer eine extra JavaScript-Datei mit dem JavaScript-Code erstellt werden. Diese wird dann im body-Bereich aufgerufen.

2. Erstellen Sie einfache Webseiten und JavaScript-Anwendungen, indem Sie die Aufgaben 6 bis 9 des Kapitels 4 im Handout bearbeiten.

3. Nutzen Sie die Aufgabe 10 des Kapitels 4, um Ihr Wissen im Rahmen der Prüfungsvorbereitung bezüglich dieses Kapitels zu überprüfen.

5. Benutzerschnittstellen testen

➢ **Sie sollen das Design, die Funktionalität und die Benutzerfreundlichkeit von Benutzerschnittstellen testen.**



Nachdem eine Software oder ein Softwaremodul fertiggestellt worden ist, wird diese Lösung getestet. Dafür stehen verschiedene Testverfahren zur Verfügung und die Software durchläuft unterschiedliche Teststufen (siehe auch Jahrgangsband 1, Modul 5). So kann beispielsweise die Funktionalität der Logik- und der Datenhaltungsschicht einer Drei-Schichten-Architektur beispielsweise durch Unittests getestet werden. Die Testdaten dafür können beispielsweise aus einer Grenzwertanalyse (siehe auch Jahrgangsband 2, Modul 8) in Rahmen eines Blackbox-Tests oder einer Pfadüberdeckung im Rahmen eines Whitebox-Tests (Kapitel 2.4.2) gewonnen werden. Nicht funktionale Anforderungen wie die Wartbarkeit von Software können u.a. im Rahmen von Audits oder Reviews überprüft werden. Aber wie wird eine Benutzerschnittstelle getestet und ist das überhaupt

notwendig? Um Antworten auf diese Frage zu erhalten, muss man die Perspektive wechseln und die Rolle der Benutzer der Software einnehmen. Benutzer haben am Anfang keine Kenntnisse über die Software und deren Funktionalität. Es ist die Benutzeroberfläche der Anwendung, die entscheidet, ob Benutzer die Anwendung weiterverwenden oder nicht. Normalerweise bewerten Benutzer zunächst das Design und das Aussehen der Anwendung und wie einfach es ist, die Benutzeroberfläche zu verstehen. Wenn Benutzer mit der Benutzeroberfläche nicht vertraut sind oder die Anwendung zu komplex ist und dadurch schwer zu verstehen, dann sind Benutzer von Anfang an ablehnend gegenüber der Anwendung eingestellt. Besonders trifft dieses auf den Bereich der Webanwendungen zu. Zum Beispiel verlassen Benutzer schnell den Webshop oder eine Buchungsseite, wenn sie sich nicht intuitiv zurechtfinden. Aus diesem Grund sollten auch die Benutzerschnittstellen ausführlich getestet werden.

Neben der Funktionalität sind vor allem das visuelle Design, die Sicherheit, die Leistung und die Benutzerfreundlichkeit Aspekte, die beim Testen genauer betrachtet werden. Wie alle Tests muss auch das Testen der Benutzerschnittstelle sorgfältig geplant werden. Im Gegensatz zu anderen Tests gibt es beim Testen von Benutzerschnittstellen weitere Herausforderungen.

Herausforderungen beim Testen von Benutzerschnittstellen

Sich ständig ändernde Benutzeroberfläche
Es ist üblich, Anwendungen ständig zu aktualisieren, um neue Funktionen und Funktionalitäten aufzunehmen. Wenn häufig Upgrades durchgeführt werden, wird die Durchführung umfassender UI-Tests zu einer Herausforderung.

Zunehmende Testkomplexität
Moderne Anwendungen verfügen teilweise über eine sehr heterogene Funktionalität wie eingebettete Frames, komplexe Diagramme, Karten und andere Elemente. Dies macht UI-Tests anspruchsvoller und umfangreicher.

Hoher und schwer zu kalkulierender Zeitaufwand
Das Planen und die Durchführung von UI-Tests können unter Umständen viel Zeit in Anspruch nehmen. In diesen Bereich fallen z. B. die Usability-Tests. Da sich die Benutzeroberfläche ständig ändert, ändern sich auch die Tests. Dies erhöht den Zeitaufwand für UI-Tests und verzögert dadurch den Bereitstellungsprozess. In vielen Fällen lässt sich der Zeitaufwand für die Tests sehr schwer kalkulieren.

Umgang mit Mängeln und Fehlern
Nicht immer lassen sich gefundene Fehler und Mängel schnell beheben. Viele einfache Mängel bei der Benutzerfreundlichkeit können zur Behebung teilweise komplexe Änderungen an der Benutzerschnittstelle erforderlich machen.

Da heutzutage die grafischen Benutzeroberflächen, sowohl bei Desktop-Anwendungen als auch im Web- und mobilen Bereich, den überwiegenden Anteil an den Benutzerschnittstellen ausmachen, wird das Testen der GUI in den Mittelpunkt den nachfolgenden Ausführungen gestellt. Dabei wird sich zum einen auf das Testen der grafischen Oberfläche selbst und zum anderen auf das Testen der Benutzerfreundlichkeit konzentriert.

5.1. Grafische Benutzerschnittstelle testen

➢ **Sie sollen grafische Benutzerschnittstellen testen und dafür Testfälle aufstellen.**



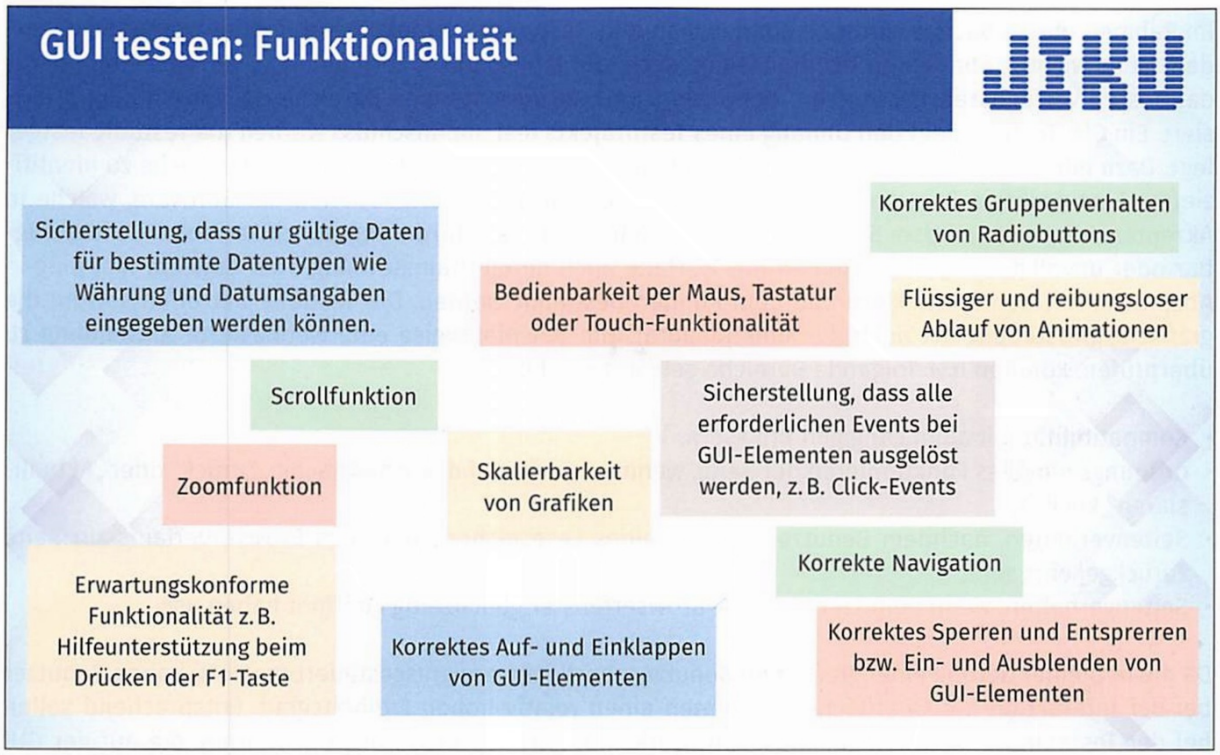
Mit GUI-Tests soll sichergestellt werden, dass eine neue grafische Benutzerschnittstelle alle geforderten Funktionalitäten enthält und alle für eine Anwendung definierten Spezifikationen eingehalten werden, indem die Dialoge, Fenster und deren Steuerelemente wie Menüs, Schaltflächen, Symbole usw. überprüft werden. Aber auch Designelemente wie Bilder und Grafiken spielen hierbei eine große Rolle, denn ein gelungenes Grafikdesign, welches die Kunden mögen und wünschen, z. B. Corporate Identity, trägt entscheidend zur Akzeptanz der Software bei. Dieses gilt sowohl für Desktop-Anwendungen als auch für Webanwendungen.

Wie bei den meisten Arten von Softwaretests kann der Umfang des GUI-Tests sehr breit gefächert sein und manuell oder automatisiert erfolgen. Dabei ist das Testen der Interaktion zwischen der Software und den Anwendern der Kernaspekt von GUI-Tests. Die Tests können dabei in zwei große Bereiche eingeteilt werden.

Ein Bereich befasst sich mit der Darstellung der grafischen Oberfläche hinsichtlich des vorgebenden Designs und prüft, wie sie den Anwendern erscheinen und ob die Vorgaben eingehalten werden. Kriterien sind dabei u.a. die Übersichtlichkeit und die Einheitlichkeit der gesamten Benutzeroberfläche, grafische Eigenschaften der Oberflächenelemente, z. B. die Ausrichtung, die Größe und die Farbe von Textbausteinen, aber auch das Einhalten der Erwartungskonformität. Hierzu gehört beispielsweise die Anzeige einer Sanduhr, wenn das Programm beschäftigt ist.



Der zweite Bereich behandelt das Testen der Funktionalität hinter den Benutzerelementen einer GUI und testet die Funktion und das Verhalten der jeweiligen Elemente. Es soll gewährleistet sein, dass die implementierte Funktion wie gewünscht fehlerfrei ausgeführt wird und dem Benutzer das zu erwartende Verhalten übermittelt.



GUI-Testprozesse können sowohl manuell als auch automatisch erfolgen. Dabei werden drei grundlegende Testverfahren unterschieden.

Testverfahren zum Testen von grafischen Benutzeroberflächen

Manuelles Testen
Beim manuellen Testen führt ein menschlicher Tester eine Reihe von Operationen durch, um zu überprüfen, ob die Anwendung ordnungsgemäß funktioniert und ob die grafischen Elemente den dokumentierten Anforderungen entsprechen. Manuelles Testen kann sehr zeitaufwendig sein und die Testabdeckung ist teilweise sehr niedrig. Die Qualität des Testens hängt hierbei sehr stark von den Kenntnissen und Fähigkeiten des Testers ab.

Aufnahme- und Wiedergabe-Testen (Record-and-Replay-Testing)
Das Aufnahme- und Wiedergabe-Testen wird mithilfe von Automatisierungstools ausgeführt. Das automatisierte UI-Testtool zeichnet alle Aufgaben, Aktionen und Interaktionen mit der Anwendung auf. Die aufgezeichneten Schritte werden dann reproduziert, ausgeführt und mit dem erwarteten Verhalten verglichen. Zum weiteren Testen kann die Replay-Phase mit verschiedenen Datensätzen wiederholt werden.

Modellbasiertes Testen (Model-Based Testing)
Das modellbasierte Testen wird auf der Grundlage grafischer Modelle, welche das Verhalten eines Systems beschreiben, durchgeführt. Diese Vorgehensweise bietet ein tieferes Verständnis des Systems und ermöglicht dem Tester, hocheffiziente Testfälle zu generieren. Es wird eine sehr hohe Testabdeckung (siehe auch Kapitel 2.4.1, S. 200) erreicht.

Es gibt verschiedene Formen des modellbasierten Testens:

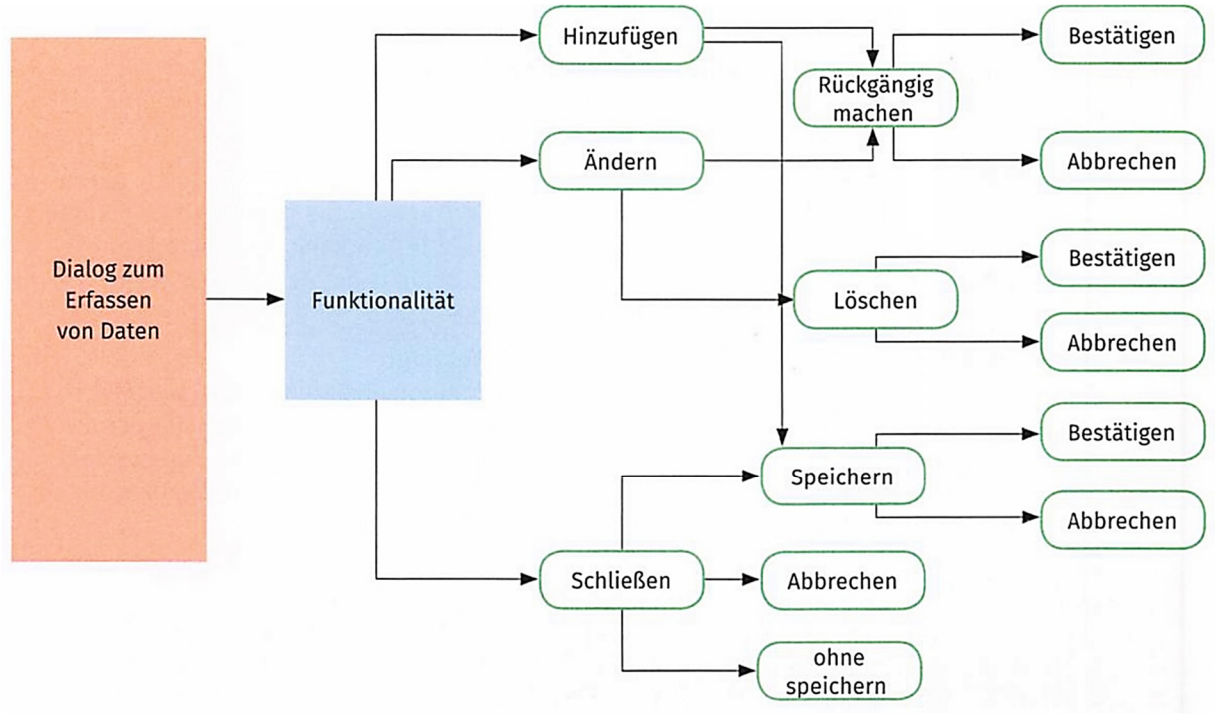
- Ereignisbasiertes Modell: Hier müssen alle Ereignisse der GUI mindestens einmal ausgeführt werden.
- Zustandsbasiertes Modell: Hier sind alle „Zustände" der GUI mindestens einmal auszuführen.
- Domänenmodell: Basierend auf der Anwendungsdomäne (abgegrenzter Einsatzbereich) der Software und ihrer Funktionalität.

Im Rahmen dieses Buches wird sich auf das manuelle Testen beschränkt. Bevor Testfälle entwickelt werden, ist es wichtig, über einen Testplan (siehe auch Jahrgangsband 1, Modul 5) zu verfügen, der die für das Testen verfügbaren Ressourcen identifiziert und die zu testenden Bereiche der Anwendung priorisiert. Ein GUI-Testplan legt den Umfang eines Testprojekts fest. Im Anschluss werden die Testfälle festgelegt. Dazu gibt es mehrere Möglichkeiten, die zu testenden Bereiche der Benutzeroberfläche zu identifizieren. Eine wichtige Grundlage dafür bieten die Anforderungsspezifikationen der Software, welche in Akzeptanzkriterien und User Storys beschrieben sind. Sind die Anforderungsspezifikationen nicht verfügbar oder unvollständig, dann können die Testfälle auch durch Brainstorming oder Concept-Mapping in Absprache mit den Entwicklern und dem Kunden bestimmt werden. Die Testfälle betreffen sowohl die grafische Darstellung als auch die Funktionalität. Um beispielsweise eine webbasierte Anwendung zu

überprüfen, könnten u.a. folgende Bereiche getestet werden:

- Kompatibilität mit allen gängigen Browsern,
- ordnungsgemäßes Funktionieren der Seite, wenn Benutzer auf die Schaltfläche „Zurück" oder „Aktualisieren" klicken,
- Seitenverhalten, nachdem Benutzer mithilfe eines Lesezeichens oder des Browserverlaufs zur Seite zurückgekehrt sind,
- Seitenverhalten, wenn Benutzer mehrere Browserfenster gleichzeitig geöffnet haben usw.

Da die Implementierung einer grafischen Benutzeroberfläche ereignisgesteuert reagiert, haben Benutzer bei der Interaktion mit Oberflächenelementen einen relativ hohen Freiheitsgrad. Entsprechend sollen bei der Testabdeckung alle möglichen Benutzerkombinationen berücksichtigt werden, die auf der GUI ausgeführt werden können. Dieses betrifft „vorgesehene" Benutzeraktionen, aber auch die „nicht vorgesehenen" Benutzeraktionen, die zu Ausnahmesituationen oder zu einem Fehlverhalten führen.



Bei einer umfangreichen GUI ist es meist möglich, über mehrere Wege einen bestimmten Zustand zu erreichen. Die Software muss auf Benutzerpfade über Tastatur, Maus oder Menüeinträge nach Vorgabe reagieren. Jedoch ist es fast unmöglich, alle Varianten der Eingaben mit Testfällen abzudecken. Hier müssen Prioritäten gesetzt werden. Dieses beschränkt auch den Testumfang.

Das Aufstellen von Testfällen soll beispielhaft an dem Dialog zur Erfassung der Daten von einem Liegeplatz vorgestellt werden.

Beispielaufgabe

Zunächst werden die Testfälle durchnummeriert und den einzelnen Oberflächenelementen zugewiesen.



Für den Dialog wurden 14 Testfälle festgelegt.

T1	Prüfung, ob die Schaltfläche gesperrt ist
T2	Prüfung, ob die Schaltfläche gesperrt ist
T3	Prüfung, ob bei nicht gespeicherten Änderungen das Speichern angeboten wird
T4	Prüfung Ausrichtung, Größe, Beschriftung Prüfung Schriftart, -farbe, -ausrichtung
T5	Prüfung Ausrichtung, Größe, Beschriftung Prüfung Schriftart, -farbe, -ausrichtung
T6	Prüfung Ausrichtung, Größe, Beschriftung Prüfung Schriftart, -farbe, -ausrichtung
T7	Prüfung Ausrichtung, Größe, Beschriftung Prüfung Schriftart, -farbe, -ausrichtung
T8	Prüfung Ausrichtung, Größe, Beschriftung Prüfung, ob die einzelnen Einträge ausgewählt werden können und ob die Liste vollständig ist
T9	Prüfung Ausrichtung, Größe, Beschriftung Prüfung, ob das Bild vollständig dargestellt wird Prüfung, ob die Skalierung der Bilder auf die vorgegebene Größe funktioniert
T10	Prüfung, ob eine Information angezeigt wird, wenn ein Eingabefeld keine Werte enthält Prüfung, ob eine Information angezeigt wird, wenn bei Länge, Breite und Wassertiefe keine gültigen Werte (z. B. Buchstaben) eingegeben wurden
T11	Prüfung, ob der Dialog zur Auswahl eines neuen Bildes angezeigt wird Prüfung, ob nach der Auswahl eines neuen Bildes dieses auch angezeigt wird
T12	Prüfung, ob bei nicht gespeicherten Änderungen das Speichern angeboten wird
T13	Prüfung der Überschrift und des Icons
T14	Prüfung, ob der Dialog nicht in der Größe verändert werden kann Prüfung, ob mithilfe der Tastatur durch den Dialog navigiert werden kann

✓ Kompetenzcheck

1. Geben Sie an, welche Aussagen richtig sind.

   a. Das Testen von GUI umfasst sowohl das Erscheinungsbild als auch die Funktionalität.
   b. Die Rechtschreibung muss sowohl bei einsprachigen als auch bei mehrsprachigen Anwendungen überprüft werden.
   c. Es sollte geprüft werden, ob die Anforderungen an die Corporate Identity eingehalten wurden.
   d. Hyperlinks müssen nicht farblich unterschieden werden.
   e. Weboberflächen sollten mit verschiedenen Browsern getestet werden.
   f. Da die Bedienung bei GUI fast ausschließlich mithilfe der Maus oder per Touch-Steuerung erfolgt, braucht die Steuerung per Tastatur nicht getestet werden.
   g. Warnungen und Fehlermeldungen sollen generell vermieden werden, um den Benutzer nicht zu verunsichern.
   h. Ereignisgesteuerte Oberflächen können eine komplexe Verhaltensstruktur aufweisen.
   i. Durch das Model-Based-Testing wird eine hohe Testabdeckung erreicht.

2. Finden Sie in Partnerarbeit weitere Beispiele für Aspekte, welche bei der grafischen Darstellung von GUI überprüft werden könnten. Diskutieren Sie Ihre Ergebnisse im Klassenverband.

3. Finden Sie in Partnerarbeit weitere Beispiele für Funktionalitäten, welche bei GUI überprüft werden könnten. Diskutieren Sie Ihre Ergebnisse im Klassenverband.

4. Bearbeiten Sie im Handout die Lernsituation 5 mit den Aufgaben 1, 2 und 3.

5.2. Das Testen der User Experience beschreiben

➢ **Sie sollen sich die Grundlagen zum Testen der User Experience erarbeiten und diese beim Testen der Benutzerschnittstelle berücksichtigen.**

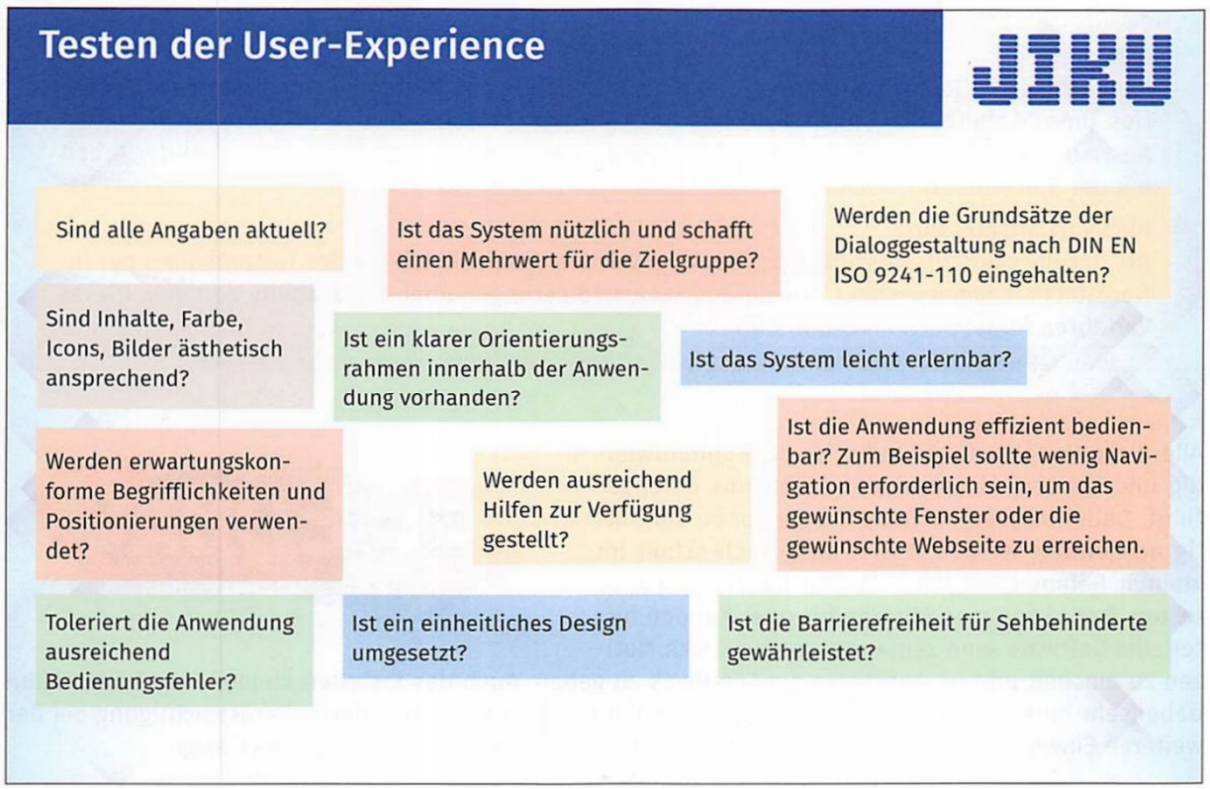


Einen weiteren Aspekt, welcher bei einer GUI getestet werden kann, ist die User Experience (S. 35). Das Testen der User Experience ist ein sehr umfangreiches Thema, welches hier nur kurz angerissen werden soll.

Bei den UX-Tests lässt man ausgewählte Versuchspersonen die Software eine bestimmte Zeit lang testen und protokolliert dabei, bei welchen Aufgaben Schwierigkeiten auftreten oder wie die visuelle Attraktivität die Software wahrgenommen wird. Dabei ist ein User-Experience-Test nicht mit einem Akzeptanztest (siehe auch Jahrgangsband 2, Modul 8) zu verwechseln. Bei einem Akzeptanztest wird überprüft, ob die Anwendung die spezifizierten Anforderungen erfüllt, z. B. dass das Drücken eines Speichern-Buttons auch die Daten speichert. Beim UX-Testen wird versucht herauszufinden, ob der Speichern-Button an der richtigen Stelle platziert wurde und eine benutzerfreundliche Bedienung ermöglicht.

UX-Tests können zu jedem Zeitpunkt in der Entwicklung durchgeführt werden und sind ein wichtiger Bestandteil im Entwicklungsprozess von Benutzerschnittstellen. Es wird empfohlen, mit UX-Tests so früh wie möglich anzufangen. So können mögliche Irritationen der Nutzer und Probleme bei der Benutzung der Anwendung frühzeitig erkannt und behoben werden.

Dabei stellt sich immer die Frage, was getestet werden soll. Hier sollten alle Punkte eines guten UX Design mit einbezogen werden. Einen großen Raum nimmt dabei die Bewertung der Usability ein. Zur Beurteilung der Usability werden in aller Regel die sogenannten „Interaktionsprinzipien" der ISO 9241-110:2020 (S. 33) herangezogen. Aus diesen Punkten lassen sich dann entsprechende Eckpunkte für das Testen ableiten.



Es gibt verschiedenen Verfahren und Vorgehensweisen für UX-Tests. Einige davon werden in der nächsten Übersicht vorgestellt.

UX-Testverfahren

Kontextanalyse
Bei der Kontextanalyse werden die Benutzer vor Ort (z. B. am Arbeitsplatz) bei der Anwendung der Software befragt und die Aufgabenbewältigung beobachtet und protokolliert.

Test mit Fokusgruppen
Bei diesem Testverfahren wird eine ausgewählte, homogene Zielgruppe von einer moderierenden Person zu Aspekten der UX befragt und dazu angehalten, Optimierungsvorschläge und Ideen für die Weiterentwicklung abzugeben.

Online-Befragung
Dieses Verfahren kommt vor allem beim Test von Webseiten zum Einsatz. Die Nutzer einer Webseite werden gebeten, an einer Online-Umfrage bezüglich der Usability der Webseite teilzuneh**men.**

Expertenbasierte Überprüfung
Hierbei überprüfen mehrere Usability-Experten anhand verschiedener Kriterien die Nutzerfreundlichkeit der Anwendung. Die Ergebnisse werden gesammelt und anschließend bewertet.

Labortest
Anders als beim Testen vor Ort wird hier eine Anwendung auf Grundlage eines speziellen Szenarios unter Laborbedingungen getestet. Dazu erfolgt im Vorfeld eine zielgruppenspezifische Auswahl an Testern. Auch hier beobachten Usability-Experten die Testpersonen bei der Arbeit mit der Software und stellen dazu Fragen.

Blickverlaufsmessung

Bei der Blickverlaufsmessung (Eye-Tracking) werden die Blickverläufe der Testpersonen per Infrarottechnik aufgezeichnet und daraus Schlussfolgerungen über die Usability gezogen. Dieses Verfahren findet auch im Labor statt.



Alle diese Verfahren sind teilweise sehr zeitaufwendig und werden von speziellen Testteams durchgeführt. Dabei müssen auch die Testpersonen sorgfältig ausgewählt werden. Es ist aber auch schon im kleinen Rahmen möglich, als Entwickler die UX zu testen. Dazu kann man Mitarbeiter oder Kunden bitten, die Software eine Zeit lang zu nutzen, sich Notizen zu machen und im Anschluss ein Feedback zu geben. Auch das Erstellen kleiner Fragebögen kann dabei sehr hilfreich sein. Das regelmäßige Einholen von Feedbacks und deren Berücksichtigung bei der weiteren Einwicklung der Benutzerschnittstelle führen am Ende zu einem guten UX Design.

Abschließend seien die Vor- und Nachteile von UX-Tests noch einmal zusammengefasst.

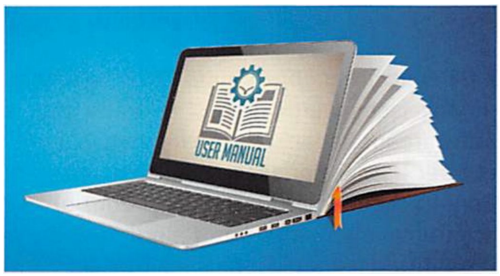
Vor- und Nachteile von UX-Tests	
Vorteile	**Nachteile**
• effizientes Softwareprodukt • Entwicklung eines Verständnisses für die Bedürfnisse der Benutzer, welche in die weitere Entwicklung einfließen können • hohe Benutzerakzeptanz der Software • Zeit- und Geldeinsparung, weil später aufwendiges Nacharbeiten vermieden wird	• hohe Testkosten • ressourcen- und zeitintensiv • aufwendige Auswahl der Testpersonen • teilweise umfangreiche Vorbereitungen notwendig • teilweise aufwendiges Auswerten der Ergebnisse erforderlich

✓ Kompetenzcheck

1. Geben Sie an, welche Aussagen richtig sind.
   a. UX-Test und Akzeptanztests sind zwei Begriffe für die gleiche Sache.
   b. Die Benutzerschnittstelle muss vollständig entwickelt sein, damit UX-Tests sinnvoll durchgeführt werden können.
   c. UX-Tests beruhen auf dem Feedback der Benutzer.
   d. Die Ästhetik der Benutzerschnittstelle spielt beim UX-Testen keine Rolle.
   e. Der Bereich der Erlernbarkeit einer Anwendung kann ein Bestandteil beim UX-Testen sein.
   f. Bei UX-Tests können Online-Befragungen zum Einsatz kommen.
   g. UX-Tests werden grundsätzlich in dafür ausgerüsteten Labors durchgeführt.
   h. Eye-Tracking ist ein Verfahren, welches bei UX-Tests angewendet werden kann.
   i. UX-Tests sind ressourcen- und zeitintensiv.
2. Überlegen Sie sich, wie und in welcher Form Sie im Arbeitskontext schon mit dem Testen der User Experience in Berührung gekommen sind. Machen Sie dazu Notizen. Diskutieren Sie Ihre Erfahrungen im Klassenverband.
3. Sammeln Sie in Partnerarbeit Pro- und Kontraargumente für das Testen der User Experience und diskutieren Sie Ihre Argumente im Klassenverband.
4. Finden Sie in Partnerarbeit weitere Aspekte, welche beim Testen der User Experience überprüft werden können. Diskutieren Sie Ihre Ergebnisse im Klassenverband.
5. Bearbeiten Sie im Handout die Aufgabe 4 des Kapitels 5 und werten Sie u. a. einen englischen Text zum Testen von UX-Design aus.
6. Überprüfen Sie Ihr Wissen über das Testen von Benutzerschnittstellen in Vorbereitung auf die Abschlussprüfung, indem Sie die Aufgabe 5 des Kapitels 5 bearbeiten.

6. Benutzerschnittstellen dokumentieren

➢ **Sie sollen sich die Grundlagen zur Erstellung von Benutzerhandbüchern erarbeiten und im Rahmen dieser Ihre Benutzerschnittstelle kundengerecht dokumentieren.**

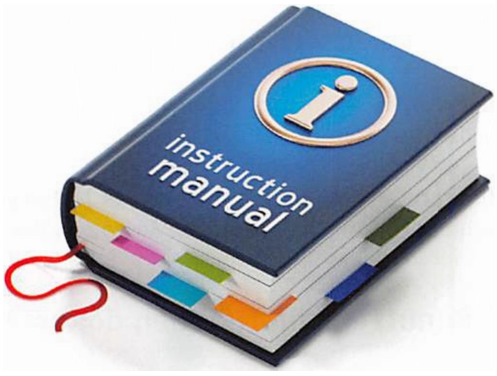


Zur Dokumentation der Benutzerschnittstellen einer Software wird in der Regel ein Benutzerhandbuch erstellt. Das Benutzerhandbuch ist ein Bestandteil einer kompletten Softwaredokumentation (siehe auch Jahrgangsband 1, Modul 5), gehört zur Anwenderdokumentation und stellt eine Sammlung von Informationen dar, welche den Benutzern einen sicheren und ordnungsgemäßen Umgang mit der Software ermöglichen.

Besonders bei der Einarbeitung in eine neue Software leistet ein Benutzerhandbuch gute Hilfe. Neben dem Funktionsumfang der Software werden dort beispielsweise die einzelnen Steuerungselemente einer GUI und deren Funktion beschrieben oder auftretende Fehlermeldungen erläutert. In vielen Fällen ist auch die Installationsanleitung noch in das Benutzerhandbuch integriert. Diese beinhaltet u.a. Informationen zu den technischen Daten der Software und die notwendigen Systemvoraussetzungen.



Es stellt sich die Frage, wie umfangreich ein Benutzerhandbuch ausfallen sollte. Der „optimale" Umfang eines Benutzerhandbuches hängt sehr stark von der Art der Software und den Bedürfnissen der Benutzer ab. Deshalb kann es hier keine einheitliche Empfehlung geben.



Bei einer kleinen Anwendung, welche auch nur von einer kleinen Benutzergruppe genutzt wird, reichen meistens schon einfache Gebrauchsanweisungen aus. Für viele solcher Softwareanwendungen ist gar kein Benutzerhandbuch notwendig, wenn z. B. die Benutzerschnittstelle selbsterklärend ist und zusätzliche Hilfen im Programm implementiert sind.

Bei einem komplexen Warenwirtschaftssystem oder einer umfangreichen Software, z. B. die Modellsoftware für das „Yachthafen Ressort", sind auf der anderen Seite sogar mehrere Bände eines Handbuches denkbar. In diesen Fällen werden meistens auch einzelne Teile, z. B. die Administration der Software, in separate Bücher ausgelagert.

Ein Benutzerhandbuch muss nicht so konzipiert sein, dass es vor der Benutzung der Software komplett durchgelesen werden muss. Es sollte eher ein Nachschlagewerk sein, welches von den Benutzern der Software im Bedarfsfall herangezogen wird. Schließlich sollten gute Benutzerschnittstellen, aufbauend auf der User Experience, möglichst selbsterklärend sein. Damit kann in vielen Fällen der Umfang eines Benutzerhandbuches erheblich reduziert werden.

Wenn der Aufbau und die Steuerung der Benutzerschnittstelle so strukturiert sind, dass diese der Denk- und Arbeitsweise der Nutzer entsprechen, die Beschriftungen und Icons eindeutig sind und bei Bedarf keine Hilfen in Form von Informationstexten in die Benutzeroberfläche integriert sind, dann benötigen Nutzer keine ausführliche Beschreibung der Oberfläche und die Dokumentation kann sich dann darauf konzentrieren, übergreifende Themen zu behandeln, Benutzer durch das Produkt zu leiten und praxisnahe Anwendungsbeispiele und Tipps zu geben. Unabhängig von dem Inhalt und dem Umfang des Benutzerhandbuches und ob es in Papierform oder digital zur Verfügung gestellt werden soll, sind immer ein paar grundlegende Punkte bei der Erstellung zu beachten.

Kriterien für ein gutes Benutzerhandbuch

Verständlichkeit
Es sollte eine einfache Sprache verwendet werden, welche den Kunden hilft, komplexe Sachverhalte zu verstehen. Ein verständliches Benutzerhandbuch trägt zu einer wesentlich kürzeren Einarbeitungszeit in die Software bei. Das Benutzerhandbuch wird für Laien geschrieben, d.h. IT-Fachausdrücke und Anglizismen sollten möglichst vermieden werden. Wenn Fachausdrücke unbedingt notwendig sind, sollten diese z. B. in einem Glossar erklärt werden.

Einfachheit
Das Benutzerhandbuch ist so einfach wie möglich zu halten. Dies gilt sowohl für den Inhalt als auch für die Gestaltung. Lange Textblöcke und Seiten, die dicht gefüllt sind mit geschriebenen und grafischen Inhalten, können dazu führen, dass Benutzerhandbücher einschüchternd wirken. Kunden bitten dann das Support-Team viel eher um Hilfe, bevor sie versuchen, ihre Fragen mithilfe des Benutzerhandbuchs selbst zu lösen. Dieser Umstand kann schnell zur Überlastung des Supports führen.

Konzentration auf das Problem
Jede Software löst ein Problem für den Kunden. Dieses Problem und dessen Lösung muss im Vordergrund stehen. In vielen Fällen werden die Kunden mit Produktinformationen überhäuft. Es wird herausgehoben, was das Produkt kann, aber nicht, warum die Benutzer dieses Produkt brauchen. Für diese ist es aber viel wichtiger zu erfahren, wie sie damit ihre Aufgaben lösen können. Diese Informationen müssen somit im Mittelpunkt stehen, d. h., es muss neben dem Wie auch das Warum beschrieben werden, z. B. warum man eine Funktion benutzt.

Fokus auf die visuelle Darstellung
Visuelle Inhalte wie Bilder, kommentierte Screenshots, Grafiken und Videos zeigen schnell, wie die Software funktioniert. Untersuchungen zeigen, dass Menschen Informationen schneller aufnehmen und Aufgaben besser ausführen, wenn Anweisungen mit visuellen oder Videoinhalten bereitgestellt werden. Visuelle Inhalte helfen auch, lange Textblöcke aufzubrechen, und können dazu beitragen, einen Großteil des Textes zu eliminieren. Beliebte Möglichkeiten, visuelle Inhalte in die Benutzerdokumentation aufzunehmen, sind Screenshots, Bildschirmaufnahmen, Tutorial-Videos usw. Diese Form der Darstellung eignet sich besonders gut zur Beschreibung der Benutzerschnittstellen.

Inhaltsverzeichnis mit klarer Struktur und logische Gliederung
Ein Inhaltsverzeichnis mit einer klaren Struktur bietet den Benutzern eine einfache, effiziente und vertraute Möglichkeit, schnell eine Lösung für ihre Frage oder ihr Problem zu finden. Es befindet sich direkt am Anfang des Dokuments, sodass sie keine Textseiten durchsuchen müssen. Es sollte alle wichtigen Überschriften und Unterüberschriften enthalten. Die Gliederung sollte einem logischen Fluss folgen. Begonnen wird dabei meist mit den einfachen Dingen und auf dieses Wissen wird nach und nach aufgebaut.

Mehrsprachigkeit
In vielen Fällen wird Software nicht nur für den heimischen Markt entwickelt, sondern auch für Länder mit anderen Sprachen. Dazu ist es notwendig, neben der Software auch das Benutzerhandbuch mehrsprachig zu gestalten. Dabei sollten die mehrsprachigen Inhalte auch mit grafischen Darstellungen der Benutzeroberfläche zusammenpassen. Idealerweise stellt man Screenshots, Grafiken oder Bildlegenden auch in der jeweiligen Sprache zur Verfügung und die Einheiten sind angepasst. Die Übersetzung sollte von professionellen und technisch versierten Übersetzern vorgenommen werden. Maschinelle Übersetzungen eignen sich auch für Benutzerhandbücher in der Regel nicht.

Früher wurden Benutzerhandbücher in Papierform erstellt. Diese Form der Benutzerhandbücher für Software ist aber zunehmend rückläufig. Trotzdem wird sie teilweise noch von Kunden aus den unterschiedlichsten Gründen nachgefragt.

Vor- und Nachteile von Benutzerhandbüchern in Papierform	
Vorteile	**Nachteile**
• kann ohne elektronisches Gerät verwendet werden • kann auf einen Blick gelesen werden • die Erfahrung, Papier in der Hand zu halten, ist einzigartig • diese Art der Leseerfahrung ist für viele angenehmer als die digitale Leseerfahrung • es kann immer und überall ohne weitere Voraussetzungen gelesen werden	• Kosten für Druck, Bindung und Verpackung • begrenzter Platz und begrenzte Informationen • kann leicht verlegt werden • nicht sofort aktualisierbar • keine Videos • hoher Aufwand, es für große Nutzergruppen zur Verfügung zu stellen

Heutzutage werden Benutzerhandbücher für Software in vielen Fällen nicht mehr in gedruckter Form zur Verfügung gestellt, sondern digital. Das bedeutet nicht, dass nicht auch eine Druckversion erstellt werden kann. Eine digitale Version aber bietet eine Reihe von Vorteilen wie Suchfunktionen, Videos oder auch eine barrierefreie Darstellung der Inhalte. Die digitalen Dateien werden entweder mit dem Produkt ausgeliefert oder das Benutzerhandbuch wird in Form einer Online-Dokumentation angeboten, welche für jeden Internetnutzer verfügbar ist oder zu der man über einen Nutzer-Account Zugang hat. Online-Dokumentationen bieten gegenüber den Druckversionen einige Vorteile.

Vorteile von Online-Dokumentationen	
Hohe Aktualität	Online-Dokumentationen können sehr schnell aktualisiert werden und sind so bei regelmäßiger Pflege immer auf dem neuesten Stand.
Hohe Multimedialität	Neben den üblichen Textbausteinen und Bildern können Online-Dokumentationen auch weitere multimediale Inhalte, z. B. animierte Grafiken und Videos, enthalten.
Hohe Verfügbarkeit	Solange ein Internetzugang besteht, bietet die Online-Dokumentation eine hohe Verfügbarkeit. Sie kann beliebig vielen Benutzern gleichzeitig zur Verfügung gestellt werden. Außerdem kann eine Online-Dokumentation nicht verlegt werden, verloren gehen oder beschädigt werden.
Bessere Interaktivität	Online-Dokumentationen bieten ausgefeilte Funktionen zur Navigation, Suche, Kommunikation und Selektion von Informationen an. Dadurch ist eine bessere Interaktivität gegeben.
Keine Beschränkungen im Umfang	Da Online-Dokumentationen praktisch keinen Beschränkungen hinsichtlich des Umfangs unterliegen, können auch Detailinformationen bereitgestellt werden. Allerdings sollte man hierbei immer die Nutzer im Auge behalten und sie nicht mit Informationen überfrachten.

✓ Kompetenzcheck

1. Geben Sie an, welche Aussagen richtig sind.

   a. Die Dokumentation der Benutzerschnittstelle ist Bestandteil des Benutzerhandbuches.
   b. Bei der Beschreibung der Benutzerschnittstelle sollten Grafiken und Screenshots im Mittelpunkt stehen.
   c. Kontaktdaten des Supports gehören nicht in ein Benutzerhandbuch.
   d. Bei der Beschreibung der Benutzerschnittstelle müssen auch eventuelle Fehlermeldungen und Warnungen erläutert werden.
   e. Für Webseiten muss immer ein Benutzerhandbuch geschrieben werden.
   f. Generell sollten Benutzerhandbücher einfach und verständlich gehalten werden.
   g. Benutzerhandbücher in Papierform bieten den Vorteil, dass auch Videos mit eingebunden werden können.
   h. Online-Benutzerhandbücher sind im Umfang nicht eingeschränkt.
   i. Bei Online-Benutzerhandbüchern ist es schwer, diese aktuell zu halten.

2. Tauschen Sie sich im Klassenverband über Ihre Erfahrungen bei der Erstellung von Benutzerhandbüchern aus und wie dieses in Ihrem Ausbildungsbetrieb umgesetzt wird.

3. Diskutieren Sie im Klassenverband die Inhalte eines Benutzerhandbuches, welche mit aufgenommen werden müssen und welche mit aufgenommen werden können.

4. Bearbeiten Sie im Handout Kapitel 5 die Aufgabe 6 und überprüfen Sie Ihr Wissen mithilfe der Aufgabe 7.