Mensch-Computer-Kommunikation

Praktikum 06

Gruppenmitglieder: Sven Boßmann
Timo Heuwer
Thomas Ihler
Johannes Theiner

Semester: Sommersemester 2018
letzte Änderung: 20. Januar 2022

Inhaltsverzeichnis

1 Heuristiken

1.1 Sichtbarkeit des Systemstatus

> Das System informiert den Nutzer immer darüber, was gerade passiert – rechtzeitig und durch angemessenes Feedback.

Bei einfachen Operationen reicht es aus dem Nutzer eine Benachrichtigung zu zeigen oder nur eine Aktivitätsanzeige darzustellen. Bei größeren Operationen ist es sinnvoll eine Fortschritsanzeige, mit weiteren Informationen wie etwa der verbleibenden Zeit, darzustellen.

1.1.1 Gute Beispiele

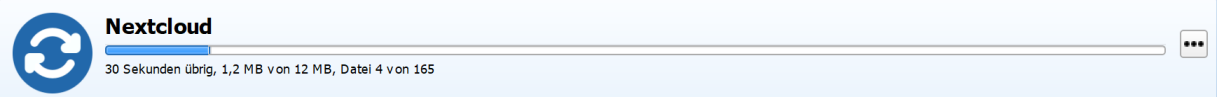

Abbildung 1: Ladebalken im Nextcloud Desktop Client

Abbildung 2: Anzeige der angeschlossenen Festplatten unter Windows, mit Anzeige des Speicherverbrauchs

1.1.2 Schlechte Beispiele

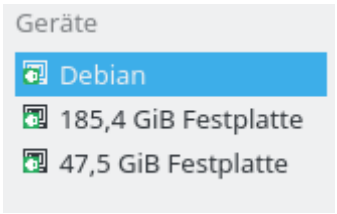

Abbildung 3: Anzeige der angeschlossenen Festplatten unter Kubuntu, wie viel Speicherplatz noch verfügbar ist ist nicht zu erkennen

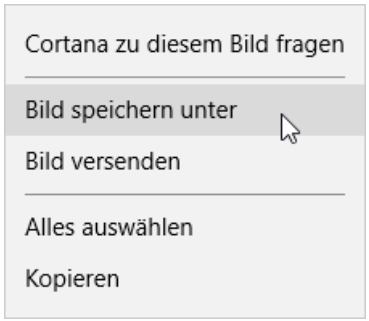

Abbildung 4: Microsoft Edge: "Bild speichern unter", ohne Benachrichtigung das der Download fertiggestellt ist

1.2 Übereinstimmung von System und Wirklichkeit

> Das System spricht die Sprache des Nutzers – mit ihm vertrauten Wörtern, Phrasen und Konzepten. Entlehnt aus der echten Welt erscheinen Informationen in ihrer natürlichen und logischen Ordnung.

Die Sprache der Anwendung sollte auf die Zielgruppe zugeschnitten sein. Bei einer größeren Zielgruppe kann es sinnvoll sein die Ausdrucksweise der Anwendung einstellbar zu machen. Auch sollten Symbole, wo möglich, den Nutzer dabei unterstützen schnell die Anwendung zu erlernen.

1.2.1 Gute Beispiele

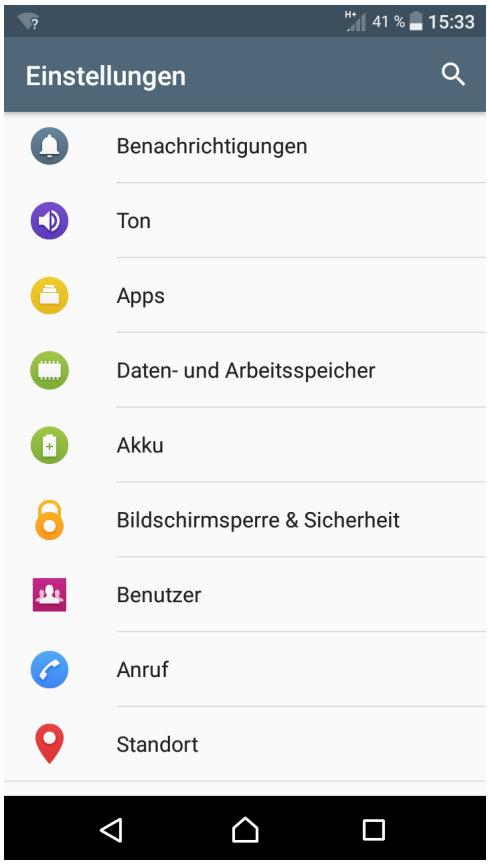

Abbildung 5: Android Einstellungen mit verständlichen Symbolen

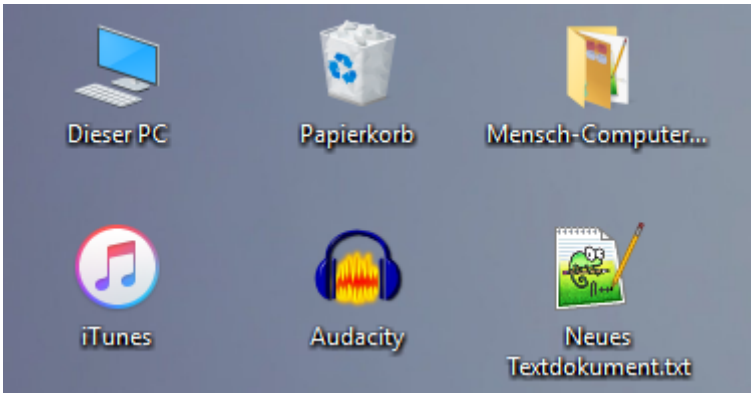

Abbildung 6: mehrere Anwendungen unter Windows mit passenden Symbole

1.2.2 Schlechte Beispiele

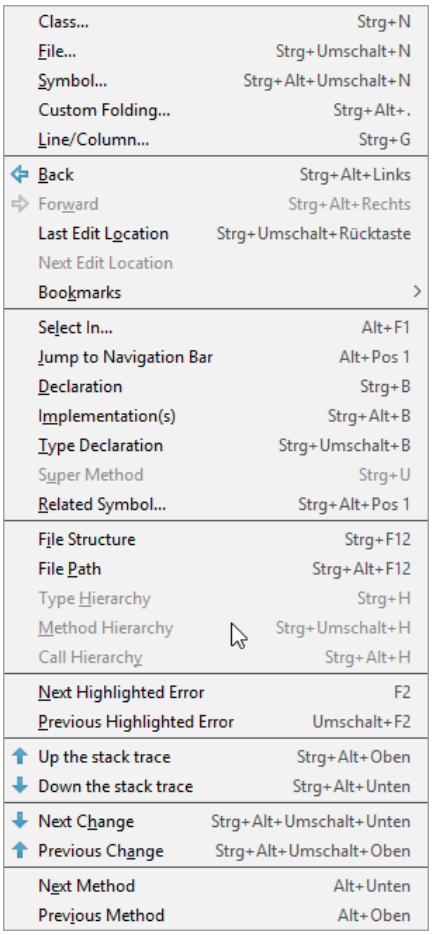

Abbildung 7: Menü in IntelliJ IDEA, nur wenige Einträge haben Symbole

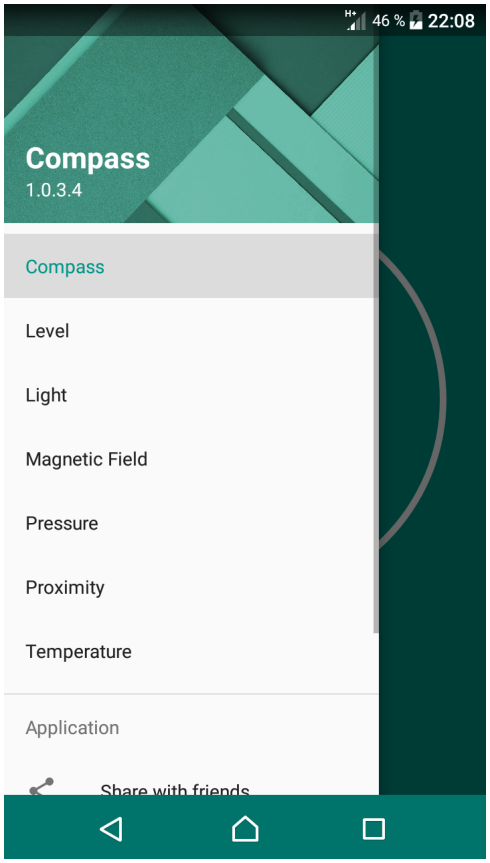

Abbildung 8: Nur ein Eintrag besitzt ein Symbol

1.3 Nutzerkontrolle und Freiheit

> Nutzer führen Aktionen oft unbeabsichtigt durch. Auswege wie „Rückgängig", „Wiederholen" und „ESC" sind deshalb immer möglich und sichtbar.

Dem Nutzer soll es möglich sein jede seiner Aktionen abzubrechen oder rückgängig zu machen. Dazu muss mindestens der letze Zustand temporär gespeichert werden.

1.3.1 Gute Beispiele

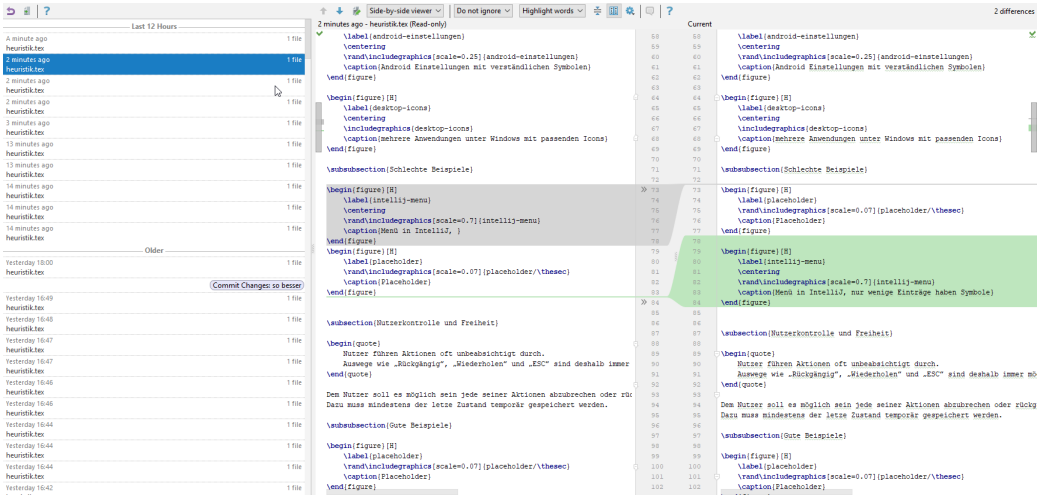

Abbildung 9: Local History in IntelliJ IDEA: jede einzelne Änderung kann nachvollzogen und rückgängig gemacht werden

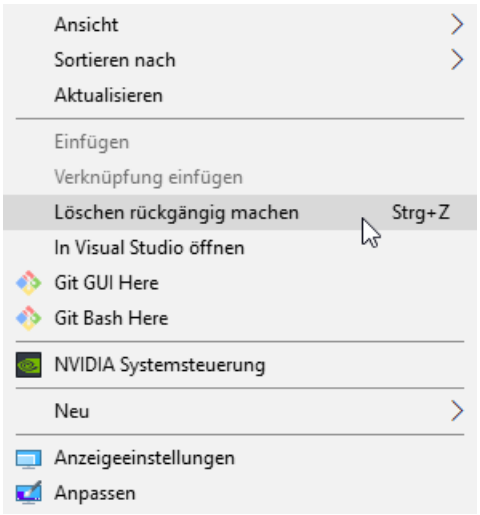

Abbildung 10: Unter Windows ist es möglich jede Dateioperation rückgängig zu machen

1.3.2 Schlechte Beispiele

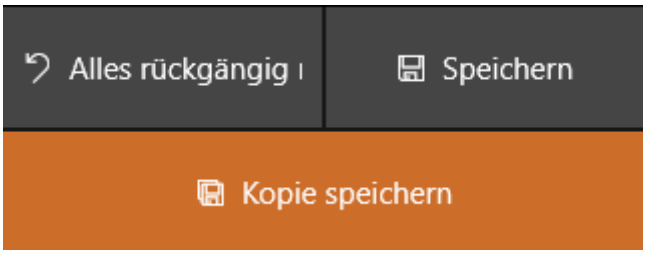

Abbildung 11: In der Windows 10 Fotos App kann das bearbeitete Bild nur auf den Orginalzustand zurückgesetzt werden, sobald gespeichert wird ist dies nicht mehr möglich

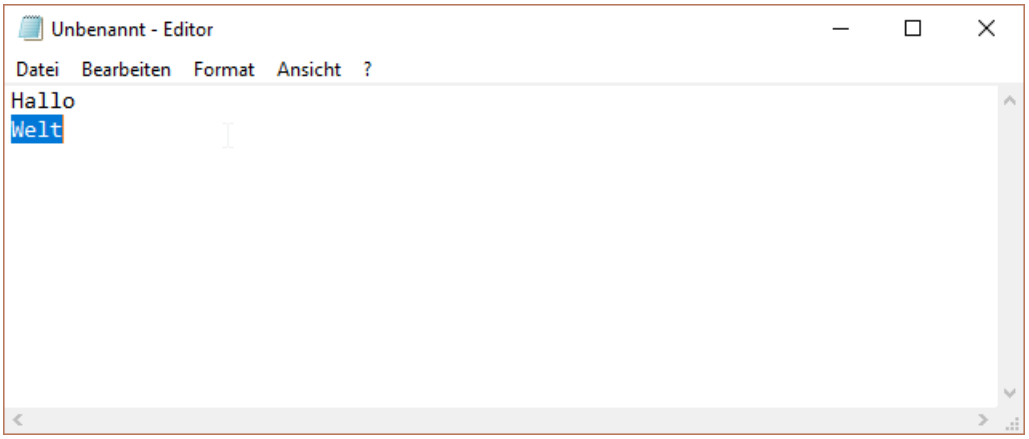

Abbildung 12: Im Windows Editor ist es kann nur die letze Operation rückgängig gemacht und wiederhergestellt werden

1.4 Beständigkeit und Standards

> Nutzer müssen nicht überlegen, ob unterschiedliche Wörter, Situationen und Aktionen das Gleiche meinen. Die Konventionen des Betriebssystems werden eingehalten.

Diese Heuristik kann erreicht werden, indem vor allem auf einheitliche Symbole und Standards gesetzt wird. Bei ähnlicher Software, wie zum Beispiel Textverarbeitungsprogrammen, wie Microsoft Word oder LibreOffice Writer, wird man feststellen, dass das Design sehr ähnlich ist. Dadurch wird gewährleistet, dass sich der User schneller zurechtfindet.

1.4.1 Gute Beispiele

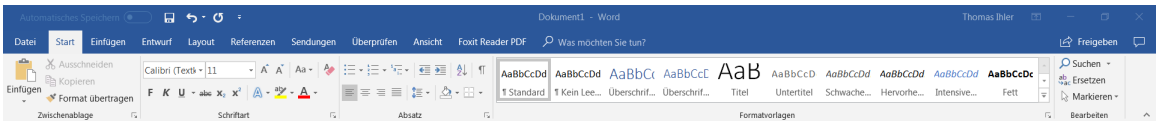

Abbildung 13: Werkzeugleiste in Microsoft Word

Abbildung 14: Werkzeugleiste in LibreOffice Writer

1.4.2 Schlechte Beispiele

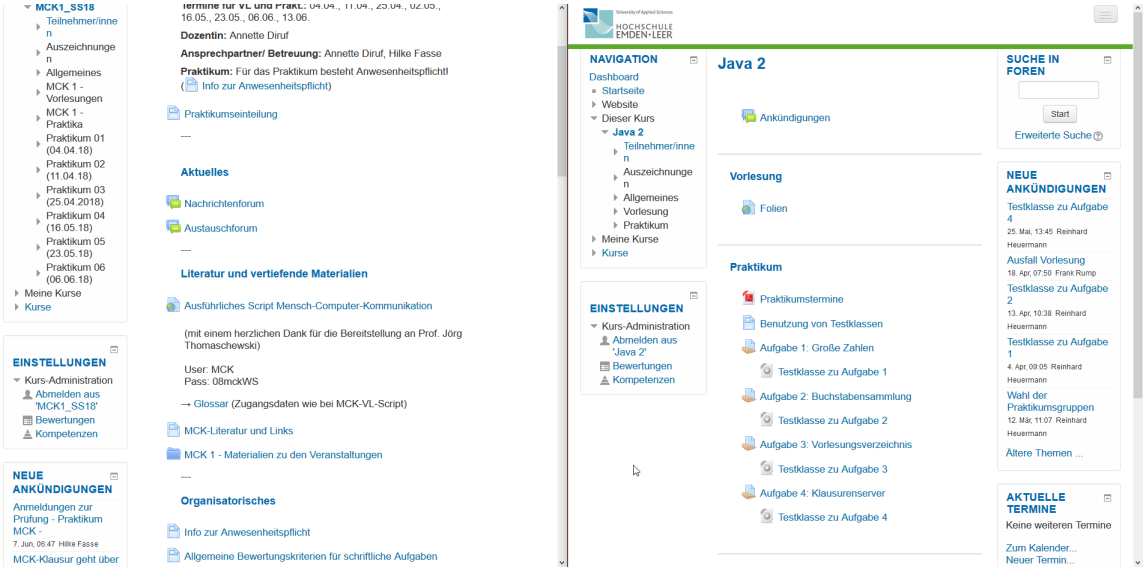

Abbildung 15: Moodle: je nach Konfiguration sind einzelne Abschnitte rechts oder links wiederzufinden

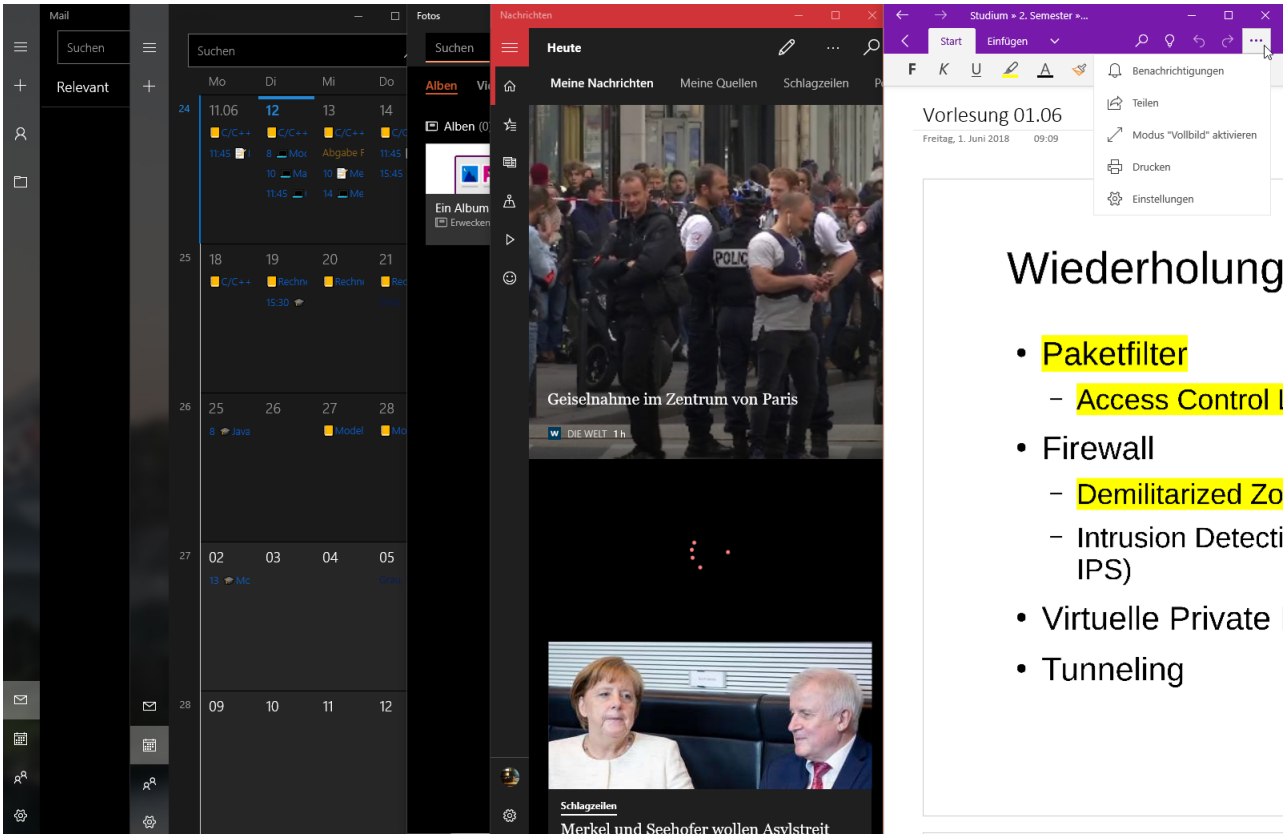

Abbildung 16: Verschiedene Windows 10 Apps: Suche und Einstellungen sind an verschiedenen Stellen zu finden

1.5 Fehlervermeidung

> Besser als jede gute Fehlermeldung ist ein sorgfältiges Design, welches Fehler gar nicht erst auftreten lässt. Das System vermeidet fehleranfällige Situationen oder warnt den Nutzer und lässt ihn die Aktion bestätigen.

Ein aufgeräumtes und organisiertes Design, trägt dazu bei, dass Fehler von Anfang an reduziert werden. Ein Beispiel wäre ein Formular, wodurch ein neuer Account auf einer Website angelegt wird. Bei vielen Websites wird schon bei der Eingabe darauf geachtet, ob diese korrekt ist. Ein weiteres Beispiel wäre es, wenn das Programm nachfragt, ob eine gewisse Aktion ausgeführt werden soll.

1.5.1 Gute Beispiele

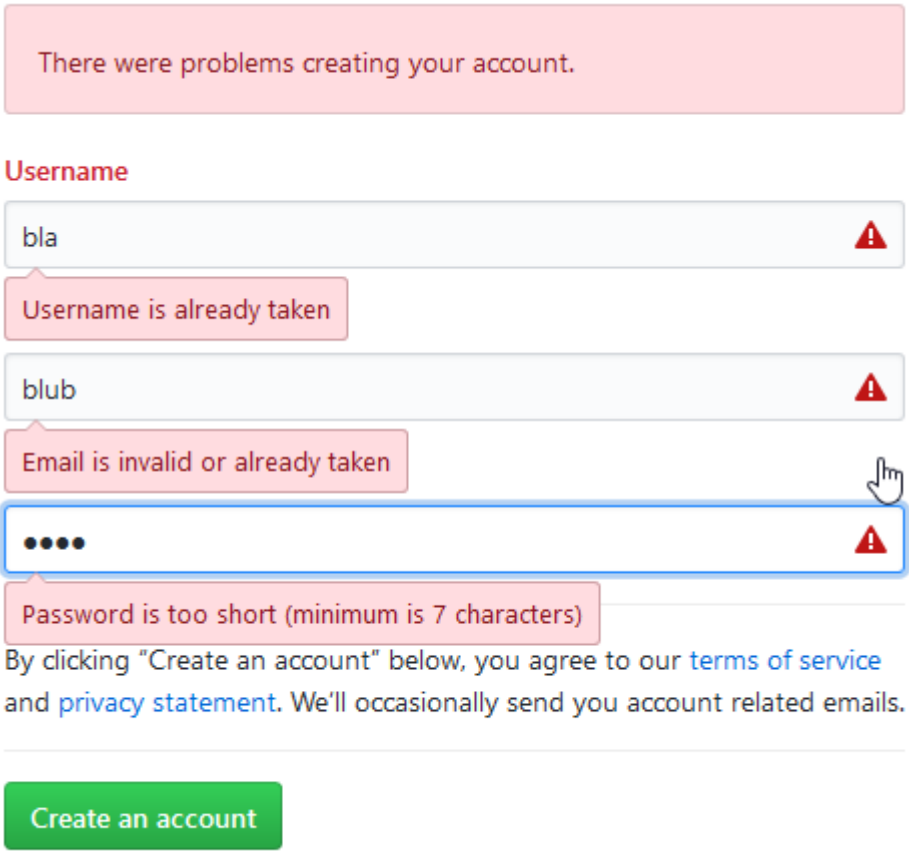

Abbildung 17: Registrierungsformular von GitHub: schon während der Eingabe wird auf Fehler überprüft

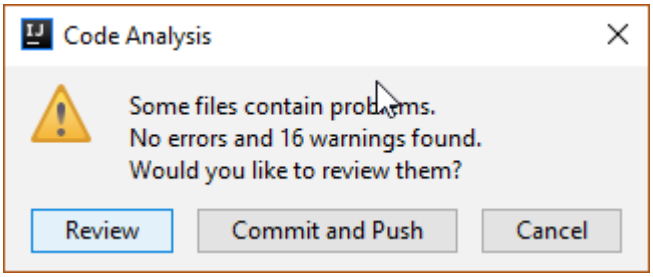

Abbildung 18: Bevor Änderungen hochgeladen werden prüft IntelliJ IDEA auf Probleme und meldet diese

1.5.2 Schlechte Beispiele

Abbildung 19: Windows Editor ohne Rechtschreibkorrektur

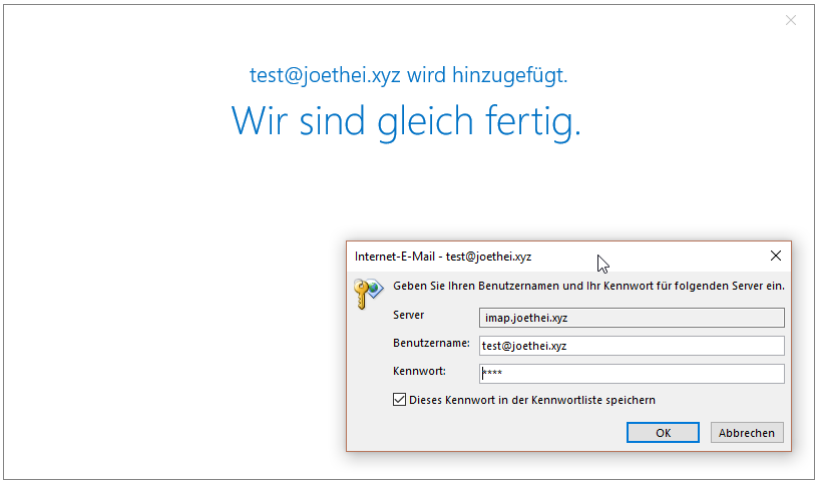

Abbildung 20: Outlook stellt auch beim Versuch eine nicht existente Email Adresse hinzuzufügen "Wir sind gleich fertig."dar

1.6 Wiedererkennung statt Erinnerung

> Durch sichtbare Objekte, Aktionen und Optionen muss der Nutzer weniger im Gedächtnis behalten. Anleitungen zum Gebrauch des Systems sind sichtbar oder leicht zu erreichen.

Dem Nutzer werden verschiedene Möglichkeiten, basierend auf bisherigen Eingaben, gegeben aus denen er wählen kann. Diese Möglichkeien können auch etwa mit einer Vorschau kombiniert werden.

1.6.1 Gute Beispiele

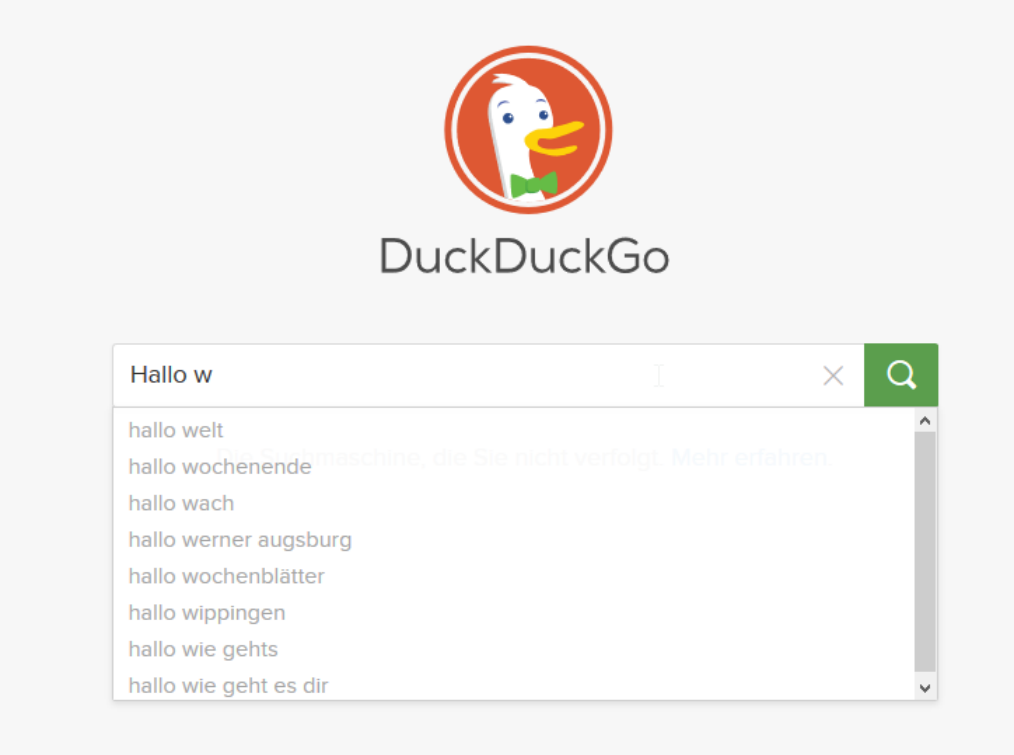

Abbildung 21: Suchvorschläge in einer Suchmaschine

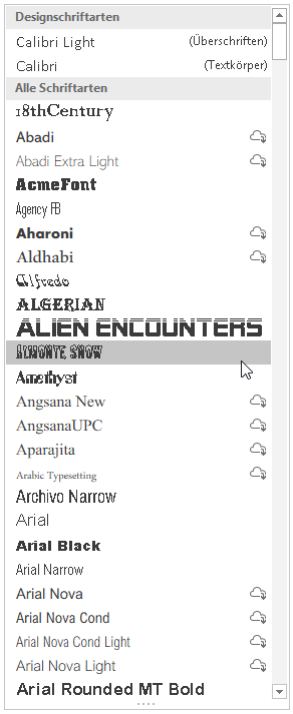

Abbildung 22: Word zeigt nicht nur den Namen einer Schriftart sondern auch wie diese aussieht

1.6.2 Schlechte Beispiele

Abbildung 23: Suchmaschine ohne Vorschläge

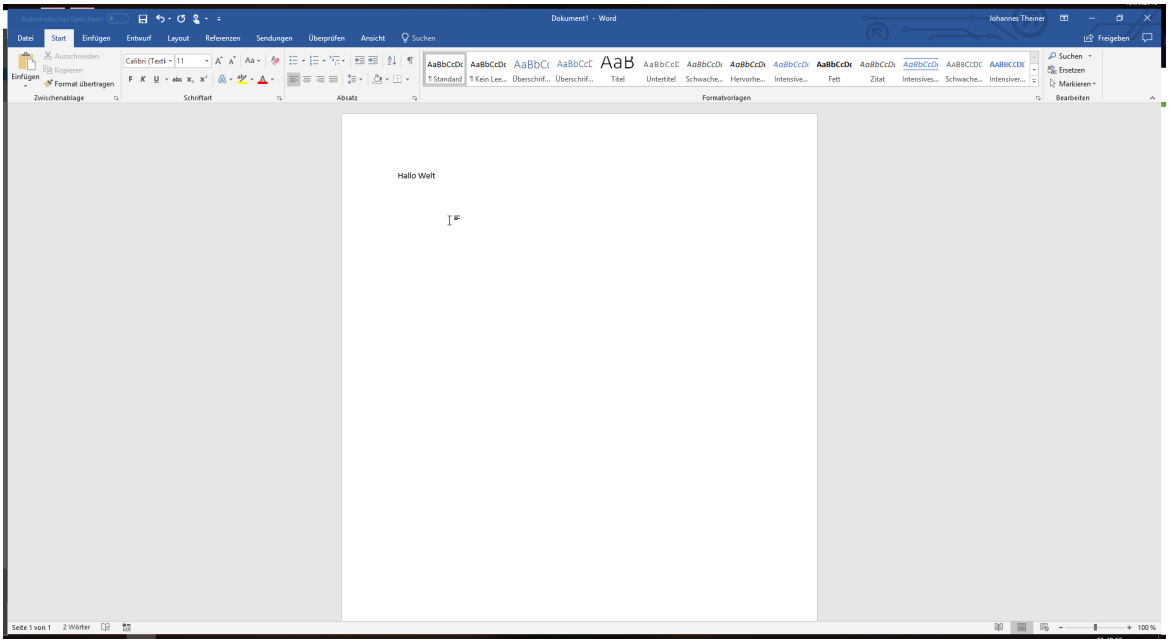

Abbildung 24: Microsoft Word, ohne Textvorschläge

1.7 Flexibilität und Effizienz

> Kurzbefehle und andere Abkürzungen – unsichtbar für Neulinge – beschleunigen bei fortgeschrittenen Nutzern die Bedienung. Zusätzlich sind häufige Aktionen individuell anpassbar.

Diese Heuristik wird erreicht, indem möglichst effiziente Shortcuts für die häufigsten Funktionen angeboten werden. Einige Programme ermöglichen auch Key-Binding, wodurch eigene Shortcuts hinzugefügt werden können. Wenn man effizient und wirtschaftlich mit einem umfangreichen Programm arbeiten möchte, ist man auf Shortcuts angewiesen.

1.7.1 Gute Beispiele

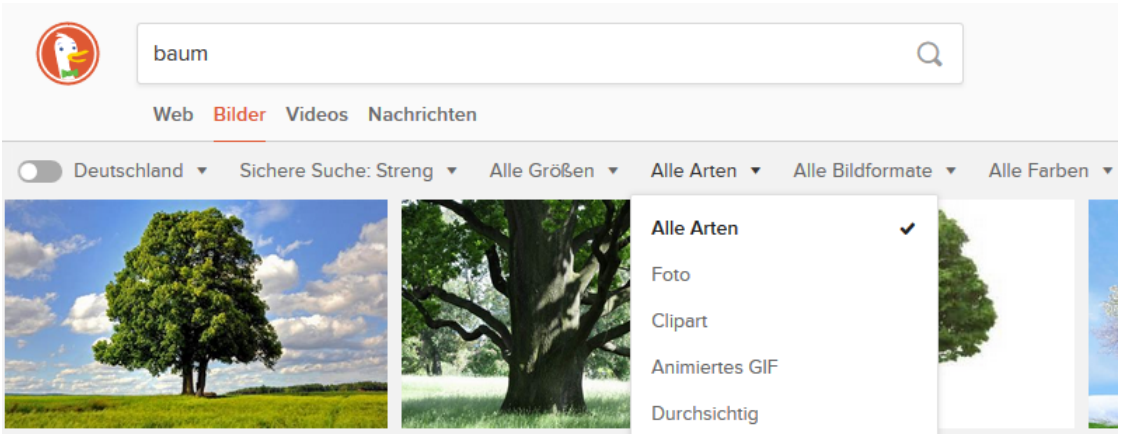

Abbildung 25: Filter erlauben wenn gewünscht eine genauere Suche

Abbildung 26: Vollständig einstellbare Shortcuts in IntelliJ IDEA

1.7.2 Schlechte Beispiele

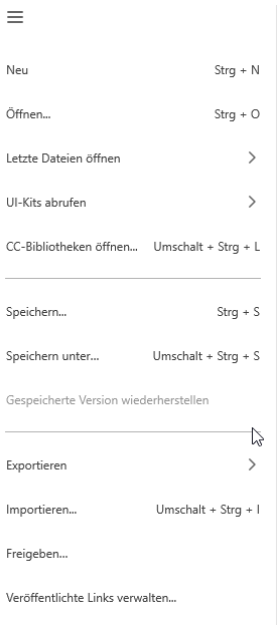

Abbildung 27: In Adobe XD ist es nicht möglich die Shortcuts anzupassen

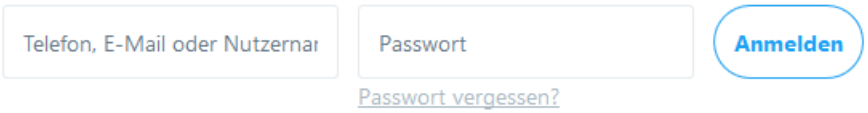

Abbildung 28: Auf Twitter springt der Cursor durch das drücken von Tab vom Nutzernamen auf "Passwort vergessen ?"

1.8 Ästhetisches und minimalistisches Design

> Dialogfenster enthalten keine überflüssigen oder nur selten gebrauchten Informationen. Denn jede zusätzliche Information steht in Konkurrenz mit den relevanten Informationen und mindert deren Sichtbarkeit.

Das Design der Seite soll möglichst simpel und minimal gehalten sein. Unwichtige Funktionalität wird auf andere Seiten ausgelagert oder ganz weggelassen.

1.8.1 Gute Beispiele

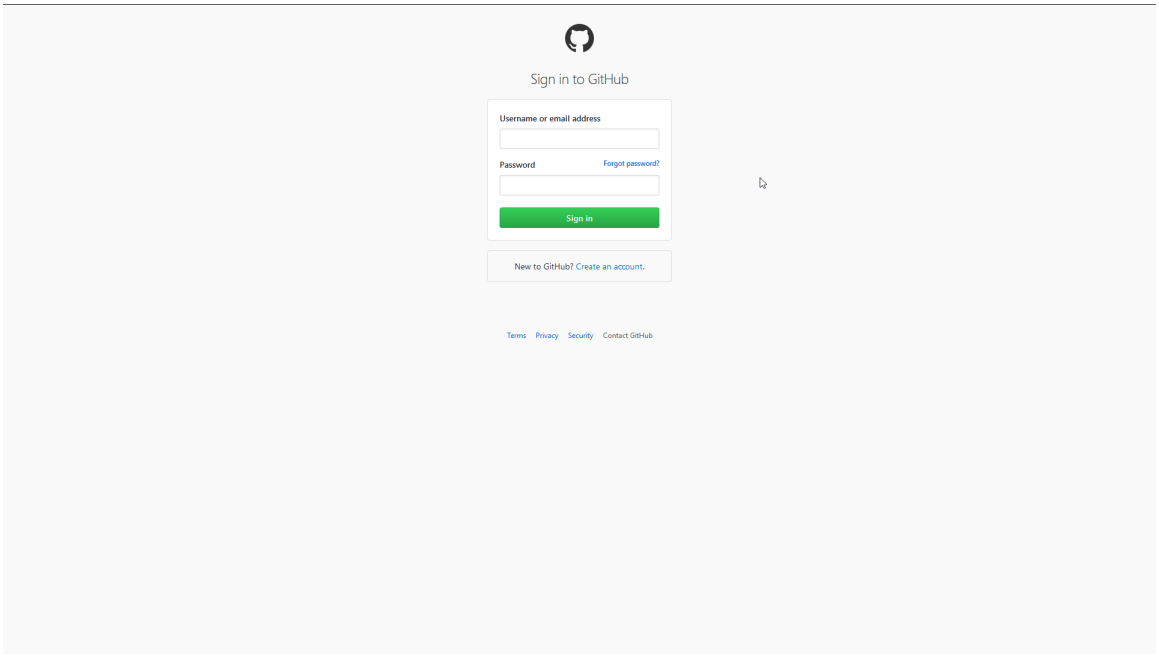

Abbildung 29: GitHub Login: nur das nötigste ist sichtbar

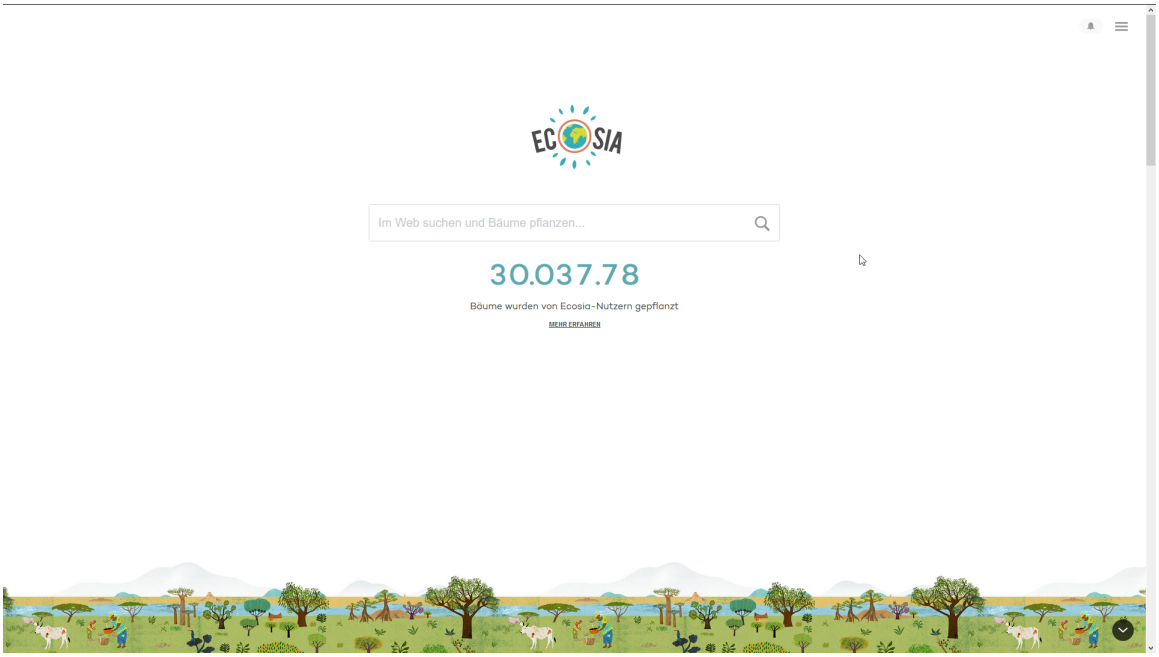

Abbildung 30: Nur das Suchfeld ist sichtbar, weiteres kann bei Bedarf über das Hambuger Menü geöffnet werden

1.8.2 Schlechte Beispiele

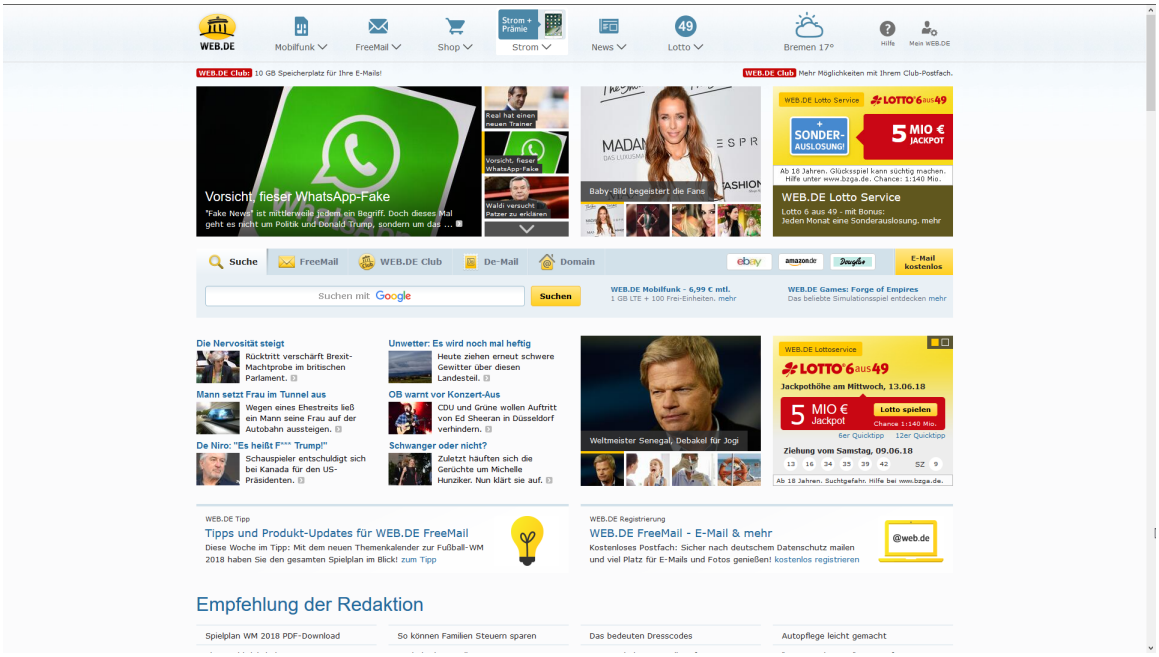

Abbildung 31: Web.de: viele Artikel und ein schwer zu findenes Suchfeld

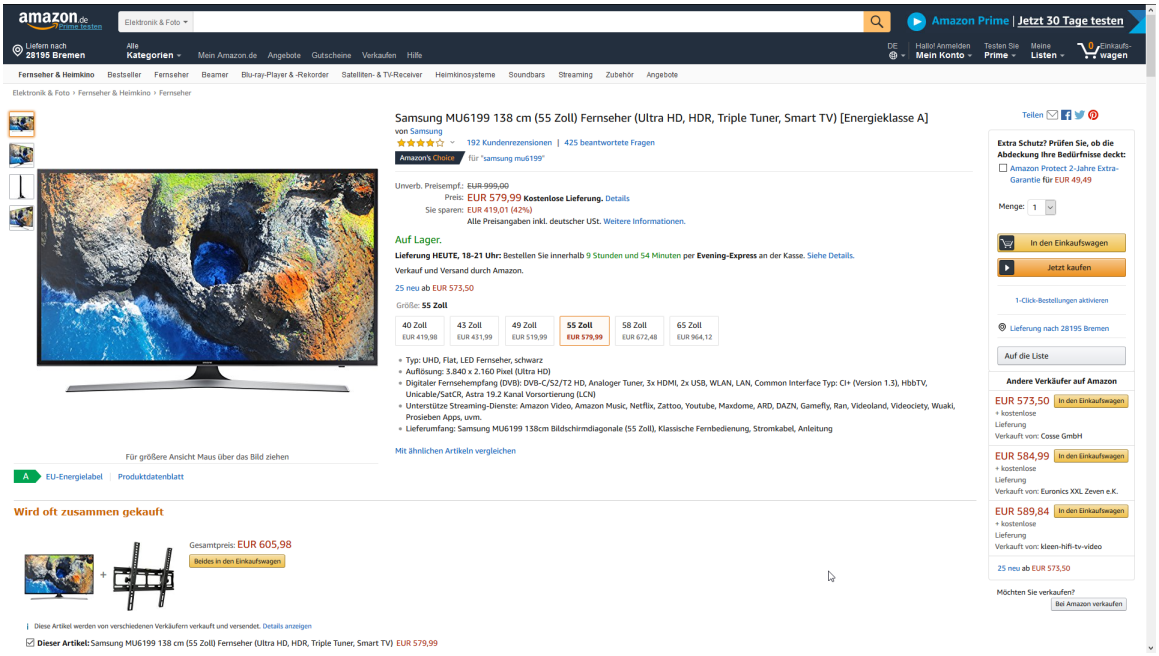

Abbildung 32: Die Seite eines Artikels auf Amazon, viele Bilder, Buttons und Texte

1.9 Hilfestellung beim Erkennen, Bewerten und Beheben von Fehlern

> Fehlermeldungen sollten in klarer Sprache (kein Code) formuliert sein, das Problem exakt beschreiben und eine konstruktive Lösung vorschlagen.

Fehlermeldungen sollten klar formuliert und aussagekräftig sein. Wenn bei jeder Fehlermeldung Fehlercodes benutzt werden, ist das nicht aussagekräftig genug. Viele Fehlermeldungen verlinken zu einer Hilfestellung.

1.9.1 Gute Beispiele

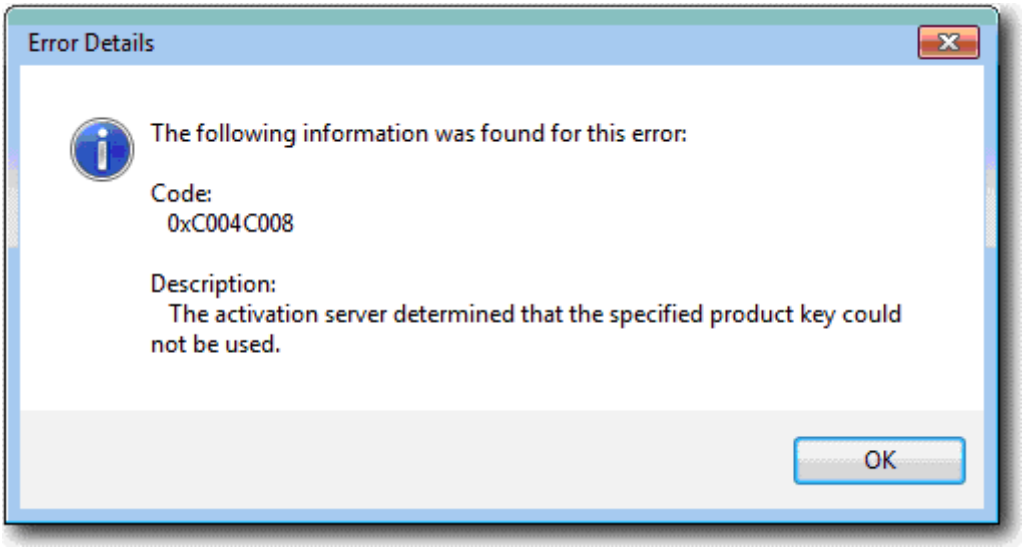

Abbildung 33: Eine aussagekräftige Fehlermeldung

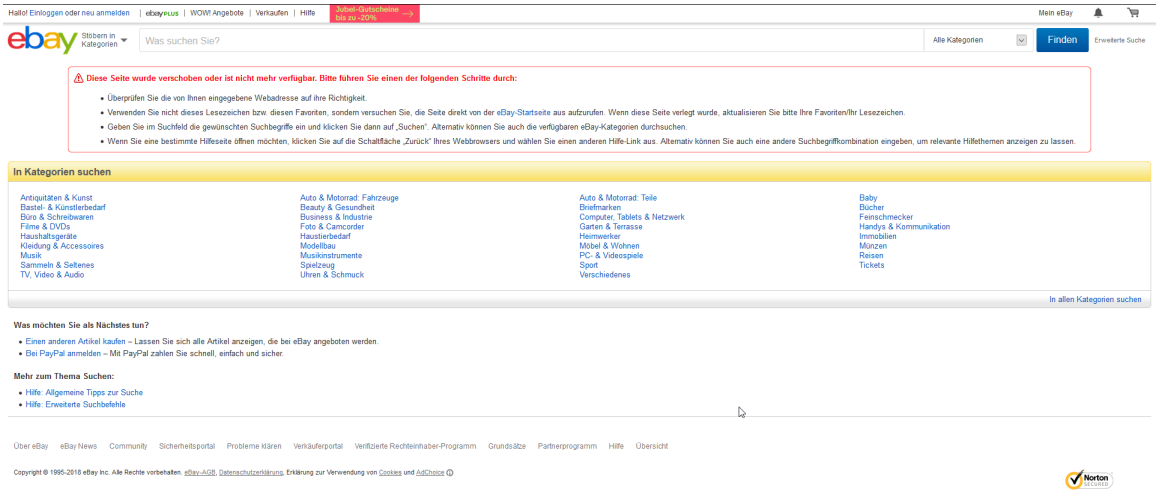

Abbildung 34: Kann eine Seite bei Ebay nicht gefunden werden werden hilfreiche Tips und Links dargestellt

1.9.2 Schlechte Beispiele

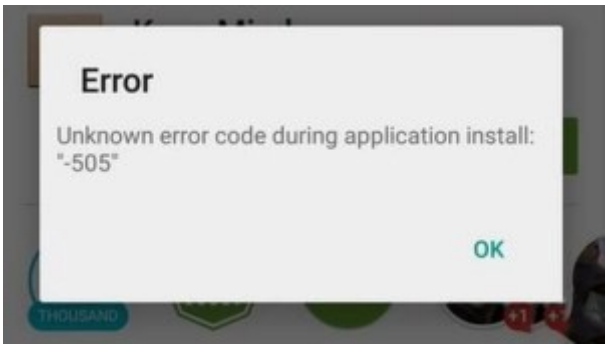

Abbildung 35: Eine kryptische Fehlermeldung

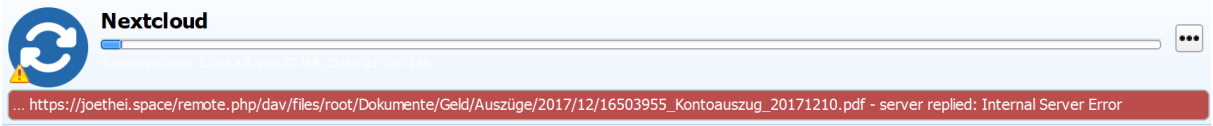

Abbildung 36: Fehlermeldung im Nextcloud Desktop Client

1.10 Hilfe und Dokumentation

> Obwohl es besser ist, wenn der Nutzer ein System ohne Hilfe benutzten kann, ist es manchmal Nötig, eine Dokumentation bereitzustellen. In dem Fall sind die Informationen einfach zu finden und konzentrieren sich auf die Aufgabe des Nutzers. Die Dokumentation enthält konkrete Schritte zur Ausführung und beschränkt sich auf das Wesentliche.

Diese Heuristik wird erreicht, wenn eine umfangreiche und hilfreiche Dokumentation vorhanden ist.

1.10.1 Gute Beispiele

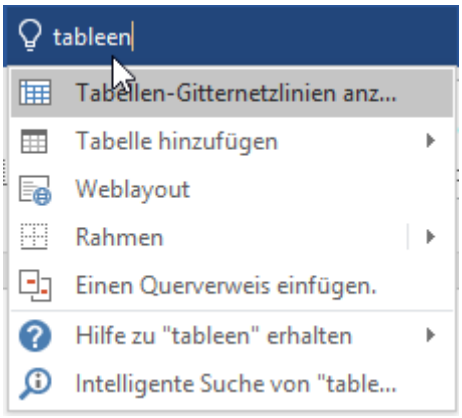

Abbildung 37: Word ermöglicht es nach Funktionen zu suchen und diese auszuführen oder die Hilfe zu öffnen

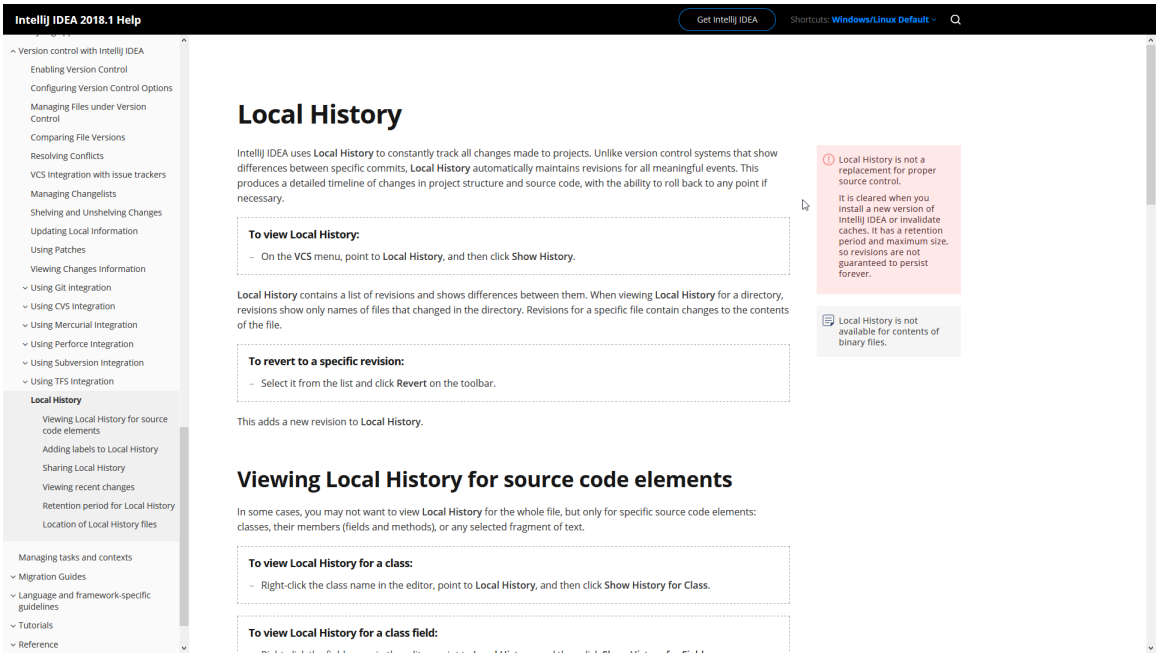

Abbildung 38: IntelliJ IDEA Hilfe: Umfangreich aber einfach und mit guter Suchfunktion

1.10.2 Schlechte Beispiele

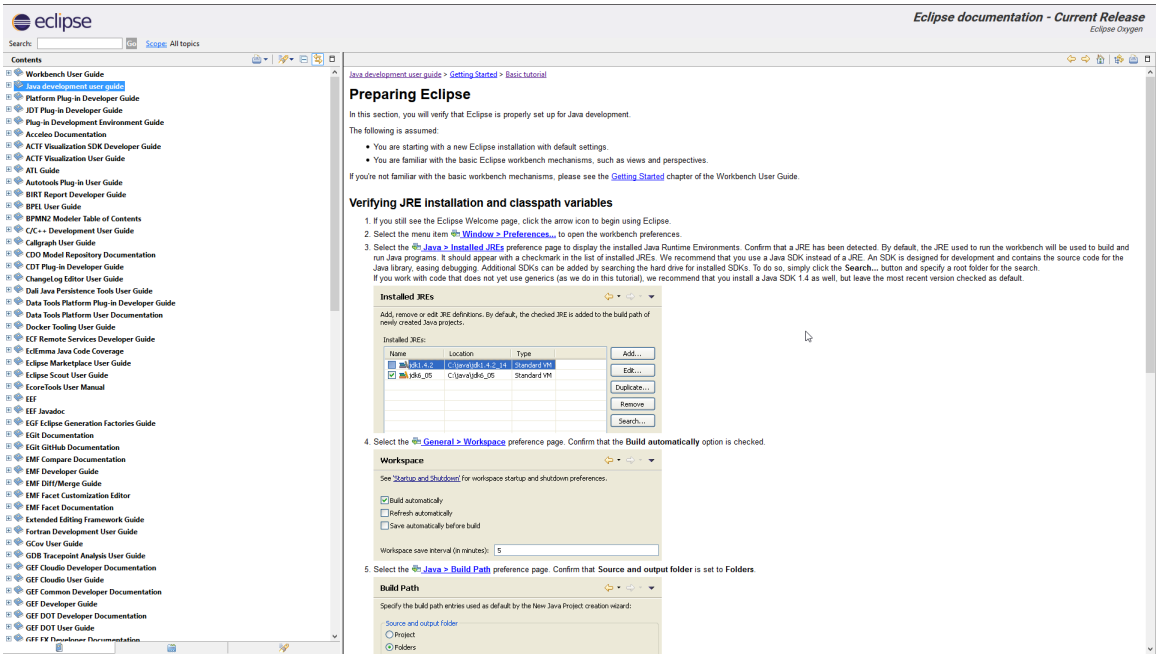

Abbildung 39: Die Eclipse Dokumentation ist unübersichtlich und nicht sinnvoll strukturiert, so finden sich etwa mehrere mehrere Einträge zu einem Thema (Wo wird es eingestellt ?, Was kann damit gemacht werden ?)

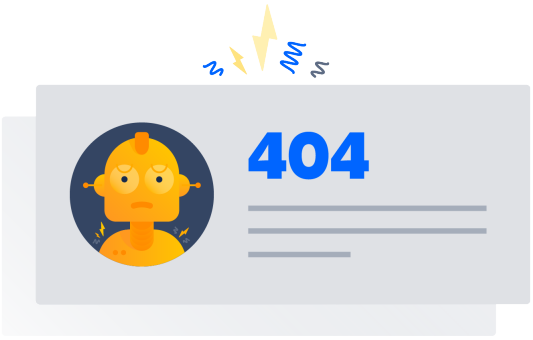

Abbildung 40: 404: Keine Dokumentation vorhanden