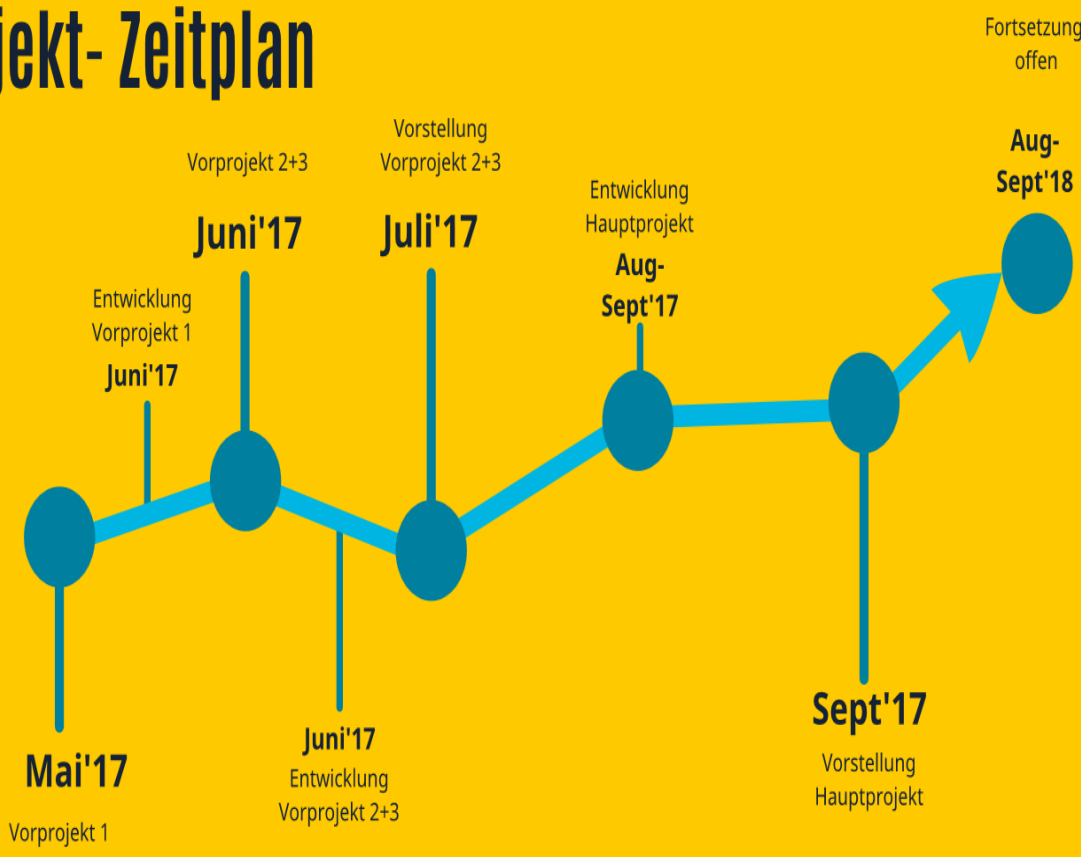
# Projektarbeit

Im Studienfach Verteilte Systeme 1
- Thema: Virtuelles Dateisystem.
- Zeitraum: SoSe 2017
- Vorgaben:
  - Netzwerkgebunden
  - 4 Grundfunktionen
  - etwa 80h Arbeitsumfang je Student.

# Projekt- Zeitplan



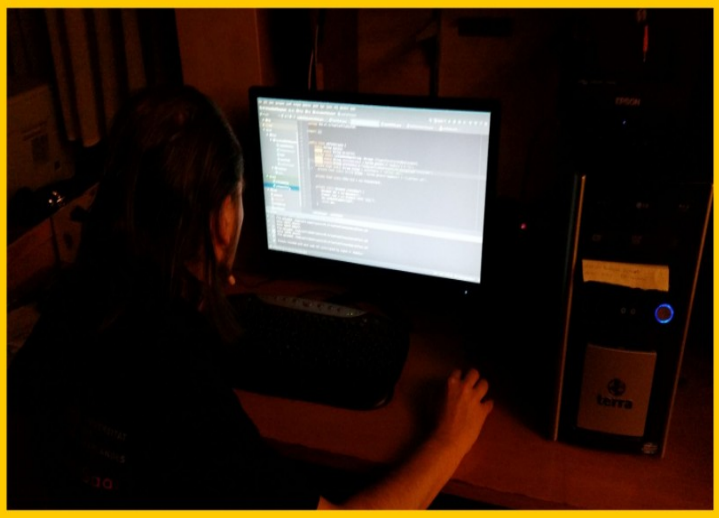
# Rahmen

Ab Beginn der Semesterferien weitestgehend in Eigenregie.
- 99,7% der Kommunikation erfolgt online via Whats App und Discord
- 99,7% des Quellcodes liegen auf dem Git-Gruppendirectory
- ab Anfang September tägliche Besprechungen ~15 Minuten täglich.
- fast tägliche Koordination und Synchronisation.
- jede Besprechung dokumentiert.

# Features

Die Software umfasst das Teilen und Synchronisieren eines Dateisystems mittels einem virtuellen Interface über eine definierte Infrastruktur

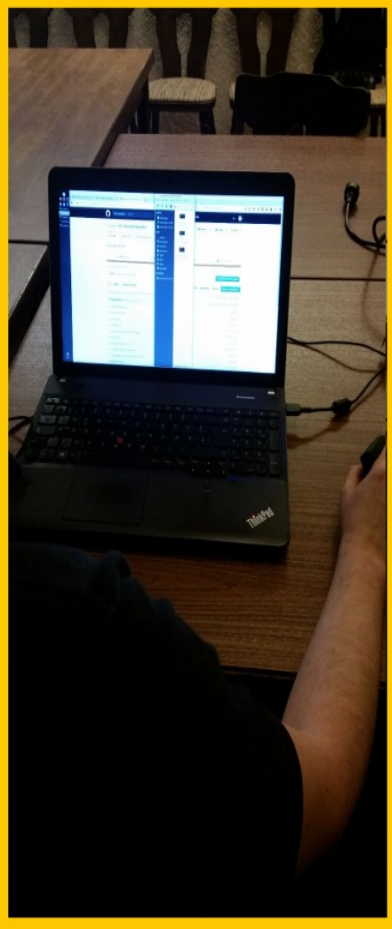
# Teilen



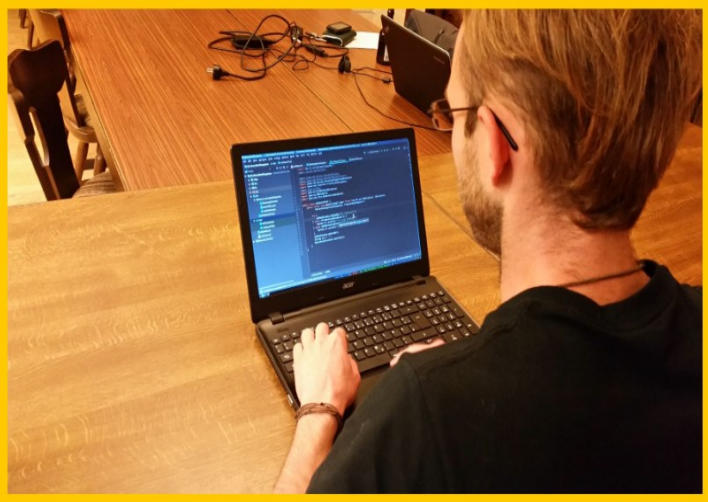
Datei- und Verzeichnis-änderungen werden binnen Sekunden von einem Filewatcher erkannt und mit Freunden, Professoren und Kollegen geteilt.

# Sync

Verknüpfen der Daten aller Teilnehmer über eine gemeinsame Datenstruktur.



# Virtuell

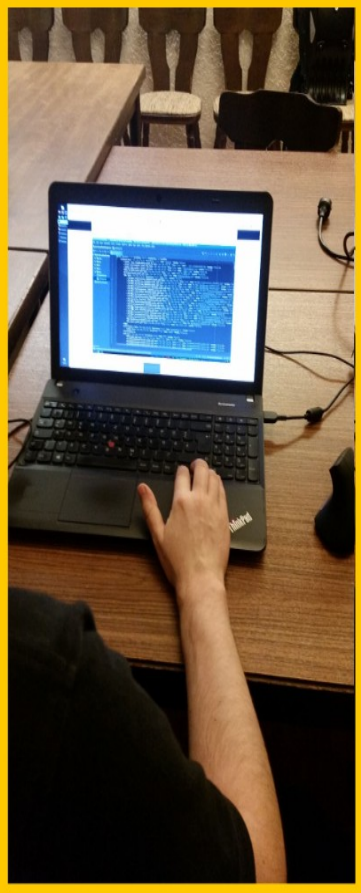


Übertragungsmedium:
Virtuelles Dateisystem

Auf dem
Speichermedium:
physisches Dateisystem

# Infrastruktur

- LAN ✓
- WAN ✗



Nutzen

# Nice2Have

für Studenten, Nerds, Power User, Blogger uvvvvvm.

Unterschied

Konkret

## Nutzen - Wofür?

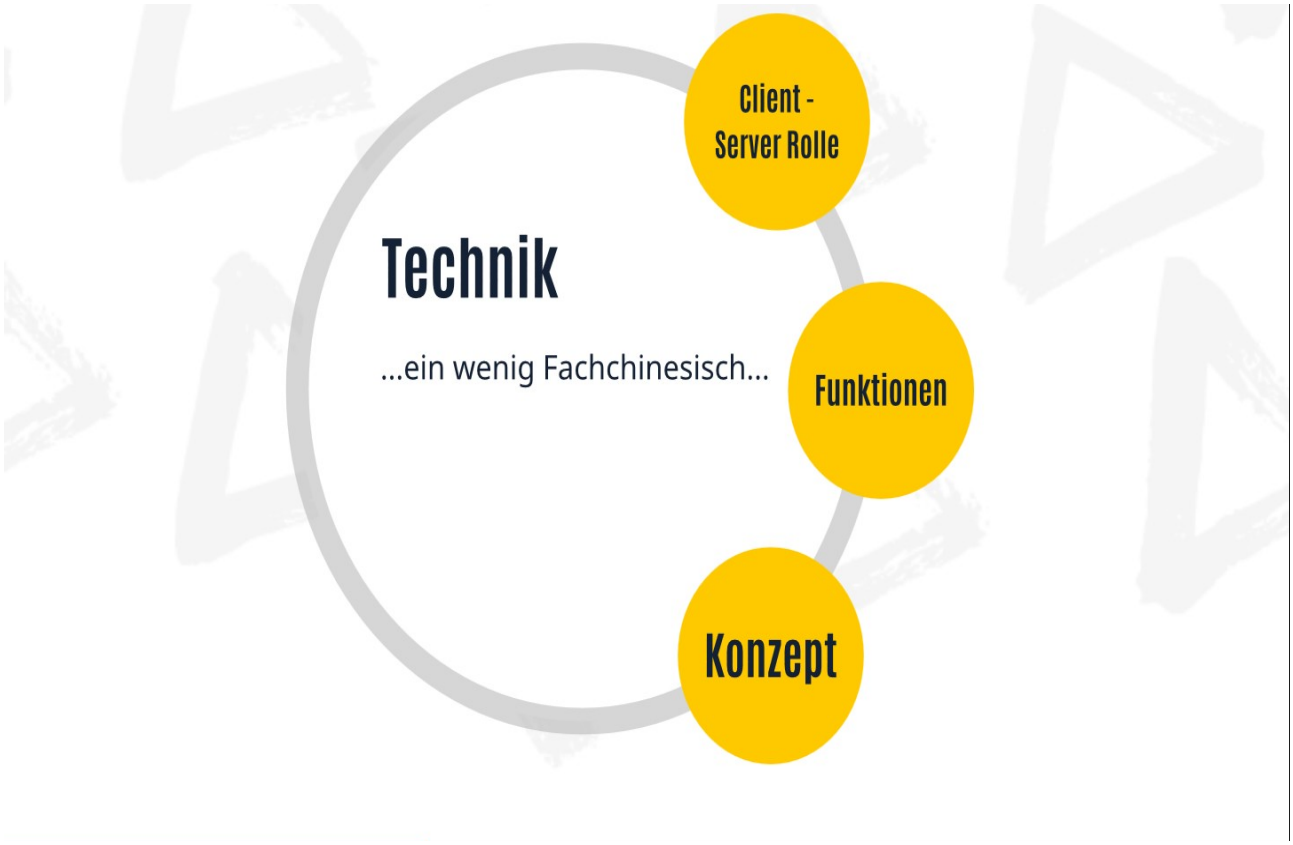
Alle Änderungen werden unter allen Clients des selben Servers synchronisiert
- Prüfungsunterlagen
- Lehrinhalte
- Messwerte
- Informationen
- Nachrichten
- Anwendungsdaten
- Projektarbeiten

## Unterschied

- schneller
- minimale Serverlast
- unkomplizierter
- plattformunabhängig
- nicht browserbasiert
- ohne Versionsnachweis
- wenn gelöscht dann gelöscht
- javabasierend
- nur Dateisystem, keine Dateien
- modular

## Konkret -Auf den Punkt gebracht...

- Jeder Client auf dem gleichen Stand
- was gelöscht wird ist weg
- betriebssystemunabhängig
- durchweg modular angelegt
- derzeit nur Konsolenanwendung




# Client - Server Rolle

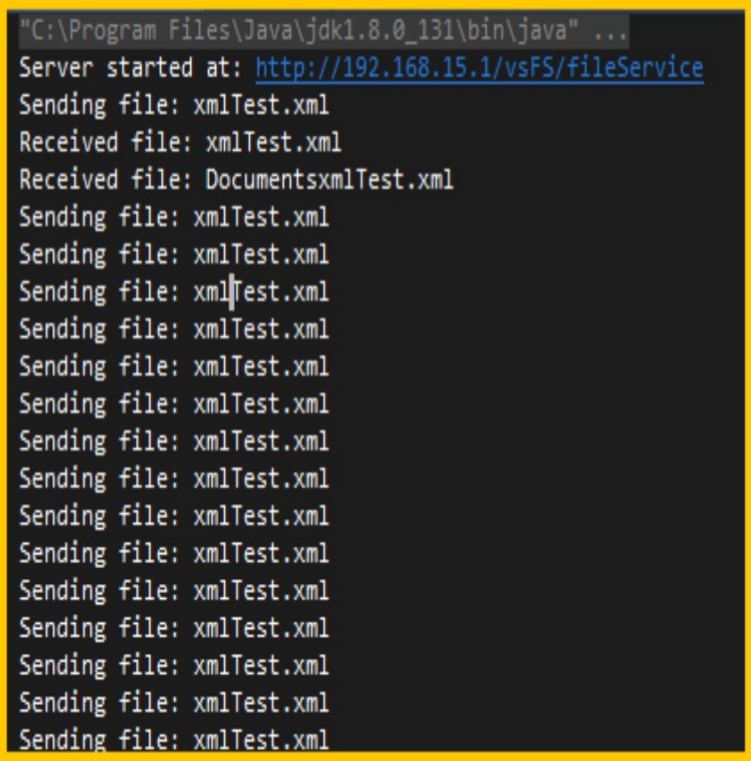
Kommunikation basiert auf dem SOAP Kommunikationsprotokoll.

Clients melden Änderungen über einen integrierten Filewatcher.

Die Clients rufen regelmäßig den Stand der angeschlossenen Clients über den Server ab.

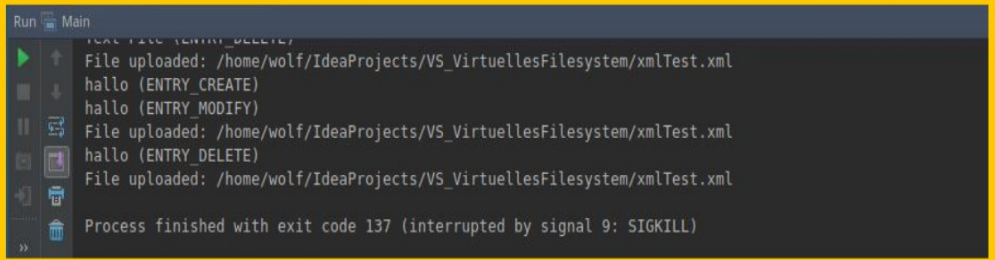
# Server

- Strukturabbild
- bietet Funktionen zum Up- und Download
- zentrale Anlaufstelle für alle Clients
- vereinfachtes Einbinden weiterer Services




# Client

- Unter einem Server zusammengefasst
- Übertragen der individuellen Dateistruktur in eine virtuelle
- synchronisieren die eigene Dateistruktur mit allen Netzwerkteilnehmern (Server puffert die aktuellste XML zwischen)
- Einfaches Filehandling auf der lokalen Maschine möglich



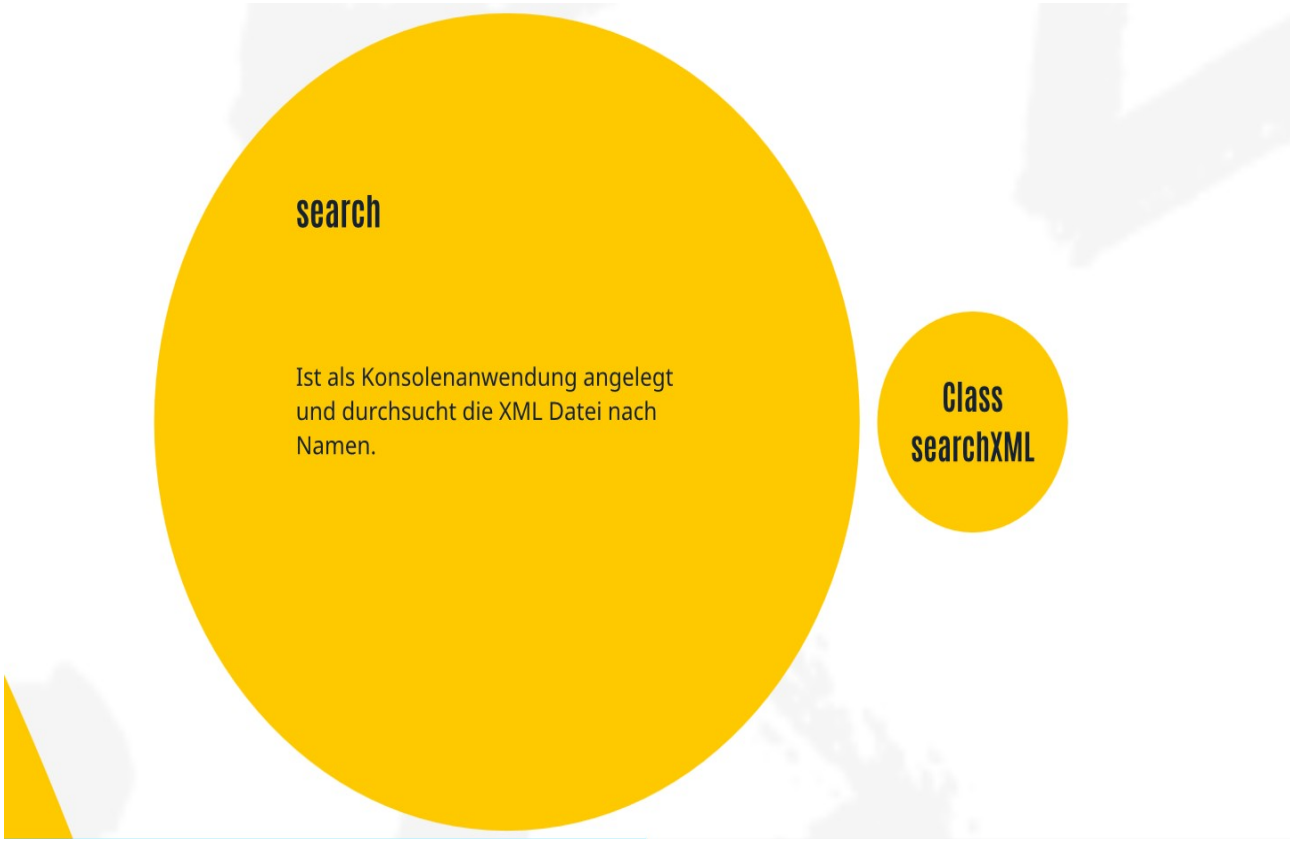
## Funktionen

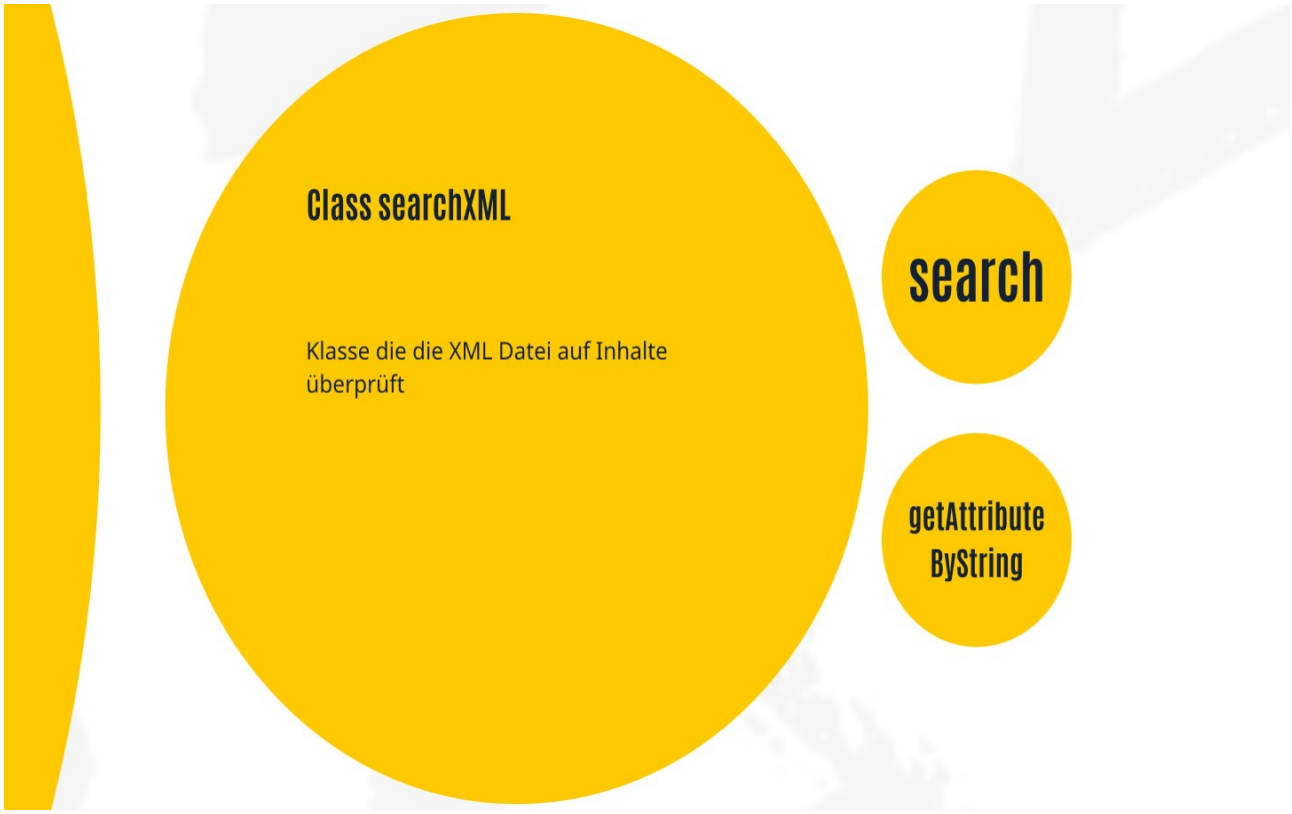
- browse
- search
- rename (move)
- delete

- xmlPathCreate
- filewatcher
- replaceIllegalCharacters


## browse

- Die Verzeichnisstruktur wird auf dem Server abgebildet und kann von dort betrachtet werden.
- Der Server "cached" quasi alle Dateiveränderungen und informiert alle bei einer ensprechenden Anfrage über die Inhaltsänderungen.
- Das Browsen selbst wird mittels Betriebssystemmitteln realisiert.


search
Ist als Konsolenanwendung angelegt und durchsucht die XML Datei nach Namen.
Class searchXML


Class searchXML
Klasse die die XML Datei auf Inhalte überprüft
search
getAttribute ByString

## search (string)

sucht in xml-Datei nach Datei/
Ordnernamen

## getAttribute ByString

zieht weitere Informationen aus der xml
Datei heraus
- node: momentaner Knoten
- string: welches Element ausgegeben
werden soll

## Beispiel

## Beispiel

<_04_BWL2_Geschäftsprozesse_0405201
6_punkt_pdf
**name**="04_BWL2_Geschäftsprozesse_04
052016.pdf"
**file**="true"
**Host**="139.19.64.94" />

## rename (move)

Das Umbenennen von Dateien und Ordnern sowie das Verschieben geschieht mittels einer lokalen Änderung und der entsprechenden Synchronisation auf bzw. mit dem Server.

## delete

Das Löschen von Einträgen die lokal erfolgt sind, werden auf dem Server abgebildet und in Folge dessen auch auf den übrigen Clients veranlasst.

## xmlPathCreate

- Verzeichnisse anlegen,
- Attribute anlegen
- Host auflösen
- (Eltern)Verzeichnisse lesen
- Unterordner erstellen
- Betriebssysteme identifizieren
- XML Datei anlegen

## createXML (string)

## detectOS()

# createXML(string)

```
public static void createXML(String dir) throws FileNotFoundException, NotDirectoryException {
    Document doc = createDoc();
    doc = writeDoc(doc, dir);
    Format format = Format.getPrettyFormat();
    format.setIndent("\t");
    try (FileOutputStream fos = new FileOutputStream(FILE)) {
        XMLOutputter op = new XMLOutputter(format);
        op.output(doc, fos);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

# detectOS()

```
public static void detectOS() {
        String os = System.getProperty("os.name");
        if (os.toLowerCase().contains("windows")) {
            DELIMITER = "\\";
        } else {
            DELIMITER = "/";
        }
    }
```

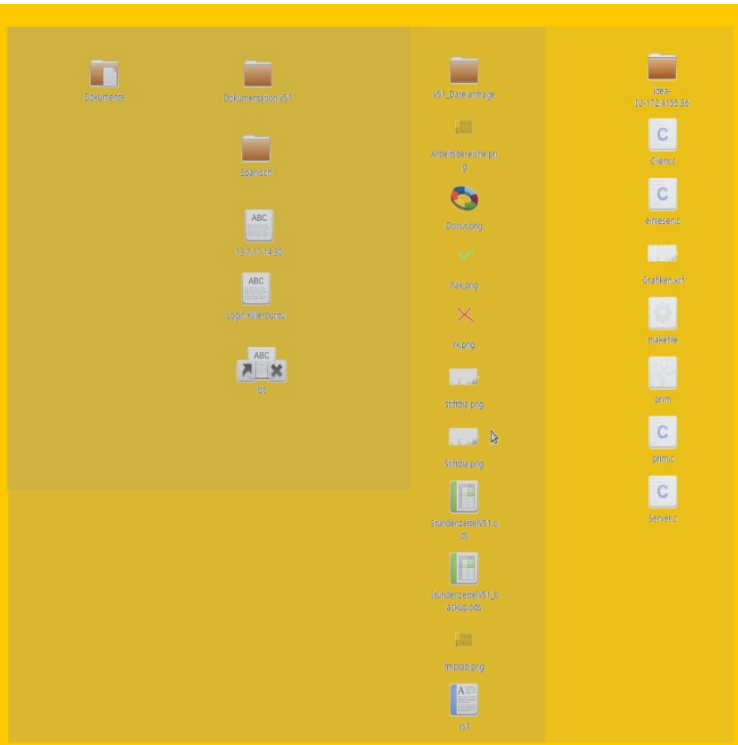
## filewatcher

- überwacht alle Ordner
- erkennt Änderungen
- pflegt Änderungen in die XML ein
- wird beim Programmstart initialisiert
- Unterordner werden durch eine Methode rekursiv mit überwacht (Eventhandler)

Beispiel

Methoden

# Beispiel



# Methoden

- startWatcher()
- initWatchkeyService()
- handleEvents(WatchServices)
- registerDirs(WatchServices)

```
    public static void handleEvents(WatchService watcher) {
//        WatchKey watchKey;
        Thread thread = new Thread(() -> {
//        while (true) {
            WatchKey watchKey = watcher.poll();
            //Wenn Ereignis geworfen wird, gebe Ereignis aus und fuehre erneut Registrierung der Ordner und Files durch
            if (watchKey != null) {
                watchKey.pollEvents().stream()
                        .forEach(e -> System.out.println(e.context() + " (" + e.kind() + ")"));
                watchKey.reset();
                registerDirs(watcher);
            }
            try {
                //An dieser Stelle das Intervall bestimmen, in dem auf Aenderungen geprueft werden soll
                Thread.sleep(200);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            //      }
        });
        thread.start();
    }
```

## replIChar

- begrenzt den Zeichensatz
- stellt den betriebssystemkonformen Umgang der Zeichenbehandlung sicher.

## Beispiel

```
private static void setIllegalCharacter() {
   illegalCharacterAndReplacement = new LinkedHashMap<>();

   illegalCharacterAndReplacement.put(" ", "-_-");
   illegalCharacterAndReplacement.put("+", "_plus_");
   illegalCharacterAndReplacement.put("[", "_sbracc_");
   illegalCharacterAndReplacement.put("]", "_sbraco_");
   illegalCharacterAndReplacement.put("=", "_eq_");
   illegalCharacterAndReplacement.put("!", "_exc_");
   illegalCharacterAndReplacement.put("#", "_hash_");
   illegalCharacterAndReplacement.put(",", "_komma_");
   illegalCharacterAndReplacement.put("$", "_dollar_");
   illegalCharacterAndReplacement.put("~", "_tilde_");
   illegalCharacterAndReplacement.put("¶", "_newL_");
   illegalCharacterAndReplacement.put("...","_ppp_");
   illegalCharacterAndReplacement.put(";","_semic_");
   illegalCharacterAndReplacement.put("†","_cross_");
}
```

## Beispiel

Host="192.168.0.104" />

<Savory_**space**__**minus**__**space**_Gutted_**punkt**_mp3 name="Savory - Gutted.mp3" file="true" Host="192.168.0.104" />

<Sudden_**space**_Death_**space**__**minus**__**space**_Party_**space**_Song_**punkt**_mp3 name="Sudden Death - Party Song.mp3" file="true" Host="192.168.0.104" />

<TenGraphs_**space**_x_**space**_Contra_space_**space**_Scandal_**space**__**minus**__**space**_The_**space**_Abyss_**punkt**_mp3 name="TenGraphs x Contra Scandal - The Abyss.mp3" file="true" Host="192.168.0.104" />

"." => "_punkt_"

## Beispiel

Host="192.168.0.104" />

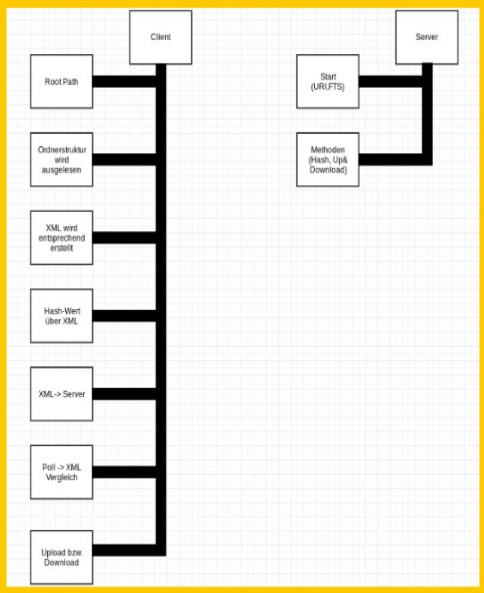
<Savory_**space**__**minus**__**space**_Gutted_**punkt**_mp3 name="Savory - Gutted.mp3" file="true" Host="192.168.0.104" />

<Sudden_**space**_Death_**space**__**minus**__**space**_Party_**space**_Song_**punkt**_mp3 name="Sudden Death - Party Song.mp3" file="true" Host="192.168.0.104" />

<TenGraphs_**space**_x_**space**_Contra_space_**space**_Scandal_**space**__**minus**__**space**_The_**space**_Abyss_**punkt**_mp3 name="TenGraphs x Contra Scandal - The Abyss.mp3" file="true" Host="192.168.0.104" />

" - " => "_**space**_-_**space**_" => "_**space**__**minus**__**space**_"

## Konzept




## Aufgaben

# Projekt -Umfang

Resumee

## Arbeitslast

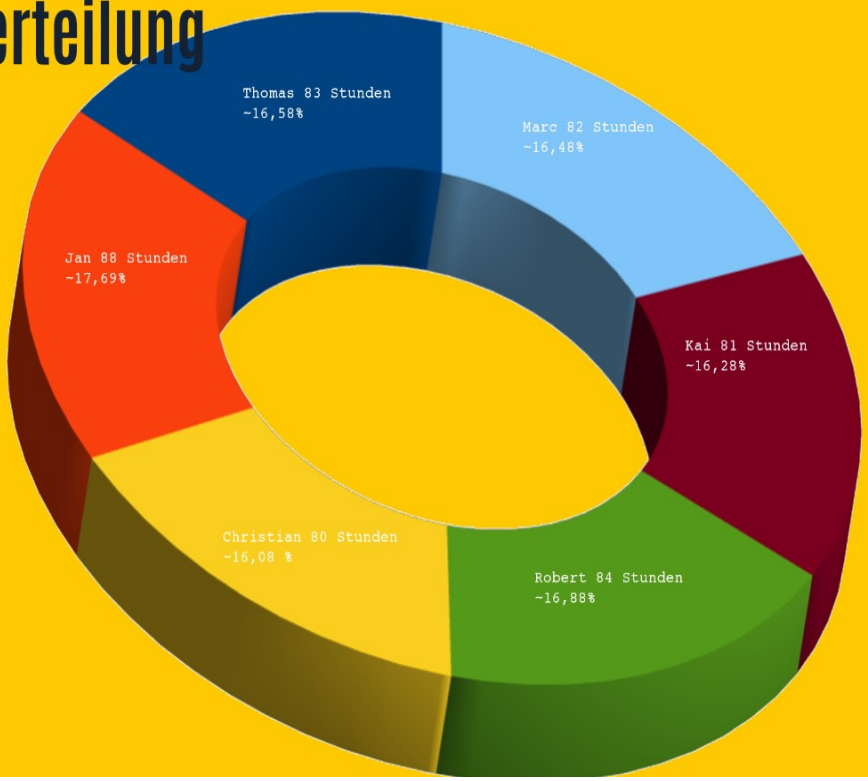
## Quellenangaben

# Aufgabenverteilung

-Hauptaufgabenverteilung-
-Nebenaufgabenverteilung-

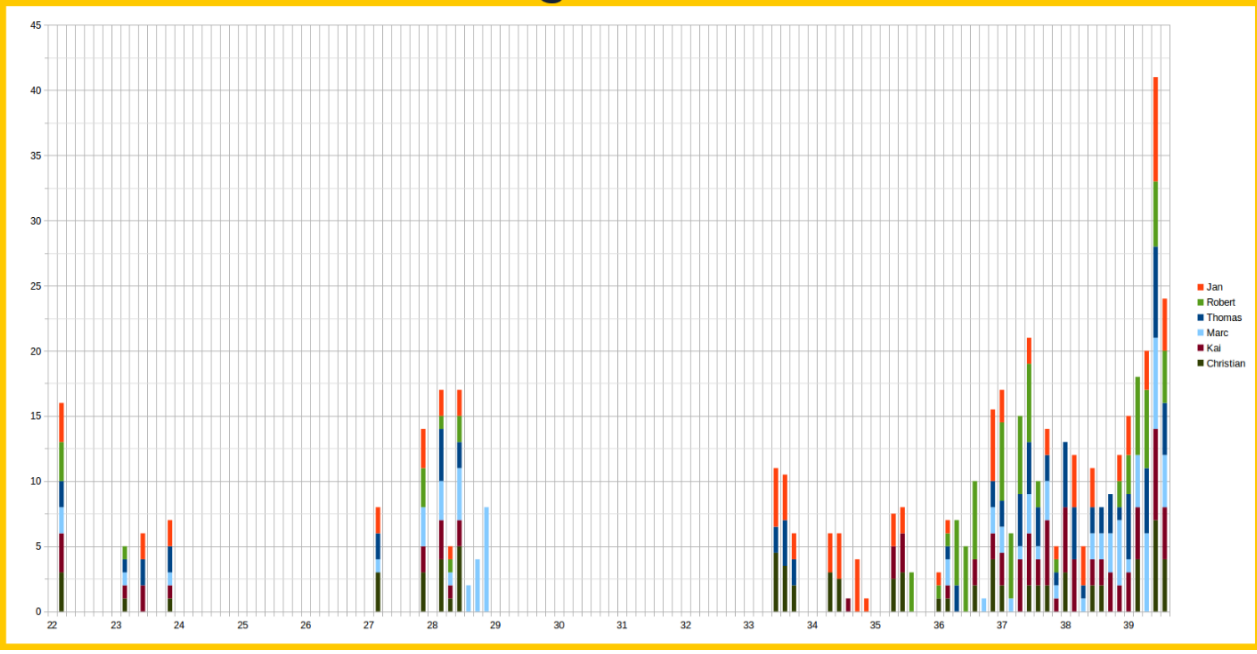
| | Funktionen | Kommunikation | Testen | Dokumentation | Präsentation | GUI | Administratives | Verzeichnisstruktur | Integration |
|---|---|---|---|---|---|---|---|---|---|
| Christian | X | | ✓ | | | | ✓ | | ✓ |
| Jan | | X | ✓ | | | | | ✓ | ✓ |
| Kai | X | ✓ | ✓ | | | | | | ✓ |
| Marc | | | ✓ | X | X | | ✓ | | |
| Robert | ✓ | | ✓ | | | | X | X | |
| Thomas | ✓ | | ✓ | | | X | ✓ | | ✓ |

# Arbeitslastverteilung




Relation: 497,5
Gesamtstunden

# Arbeitslastverteilung auf die Kalenderwochen




# Quellenangaben

incl. Informationsquellen, Hauptquellen fürs Bugfixing etc.

- Vorlesungsunterlagen Verteilte Systeme 1, Softwaretechnik 2
- Wikipedia.de
- Stackoverflow.com
- Youtube.de
- Gimp 2.8 Einstieg+Praxis
- docs.oracle.com/javase/9/

# Entwicklungsumgebung

IDE:
Windows: Intellij
Linux: Intellij IDEA (Jetbrains)

Versionskontrolle:
Github.com

Kommunikation:
Whatsapp-Gruppe, Discord-Gruppe, diverse TS Grüppchen

UML:
gliffy.com