Fachhochschule Kaiserslautern | University of Applied Sciences

VERTEILTE SYSTEME(861171,860661)

{

System.out.println(" Arman Neysi, Aryan Layes ");

}

Inhaltsverzeichnis

1. CMS Konzept

- Realisierung mit JAX-WS
- Benutzerverwaltung, Administration
  - Registration: Benutzer ist nach Registrierung noch gesperrt - keine Datei/Kalender zugriff; kann aber chatten
  - Aktivierung: Benutzer mit administrativen Rechten können gesperrte Registrationen aktivieren

1. CMS Konzept

- Die Dateiverwaltung ermöglicht:
  - Anlegen von neuen Dateien
  - Verwendete Dateien für andere User sperren
  - Bearbeiten und Löschen von nicht verwendeten Dateien

1. CMS Konzept

- Chat
  - Clients sollen untereinander chatten können, ohne dabei die Nachrichten über den WebService zu transportieren
  - Auf jedem Client läuft ein TCP Chat Server, welcher auf eingehende Verbindungen wartet
  - Client A (möchte chatten) erhält IP des Client B von Webservice
  - Nach Verbindung Client-to-Client (p2p), Webservice wird nicht mehr benötigt

2. WebService

- Implementiere Funktionen
  - Anmeldung
  - Registrierung
  - Dateiverwaltung
  - Kalender
  - Administration
  - Speicherung von Benutzerdatenbank und Kalender
  - Logging

2. Consolen Client

- Implementiere Funktionen
  - Anmeldung
  - Registrierung
  - ~~Dateiverwaltung~~
  - Kalender
  - Administration
  - Chat

2. GUI Client

- Implementiere Funktionen
  - Anmeldung
  - ~~Registrierung~~
  - Dateiverwaltung
  - Kalender
  - ~~Administration~~
  - Chat

2.1 Server - Klassen Struktur

Server.java

- Interface, benötigt für die Realisierung durch JAX-WS
- Veröffentlicht Methoden durch Annotation (Endpoint Interface)

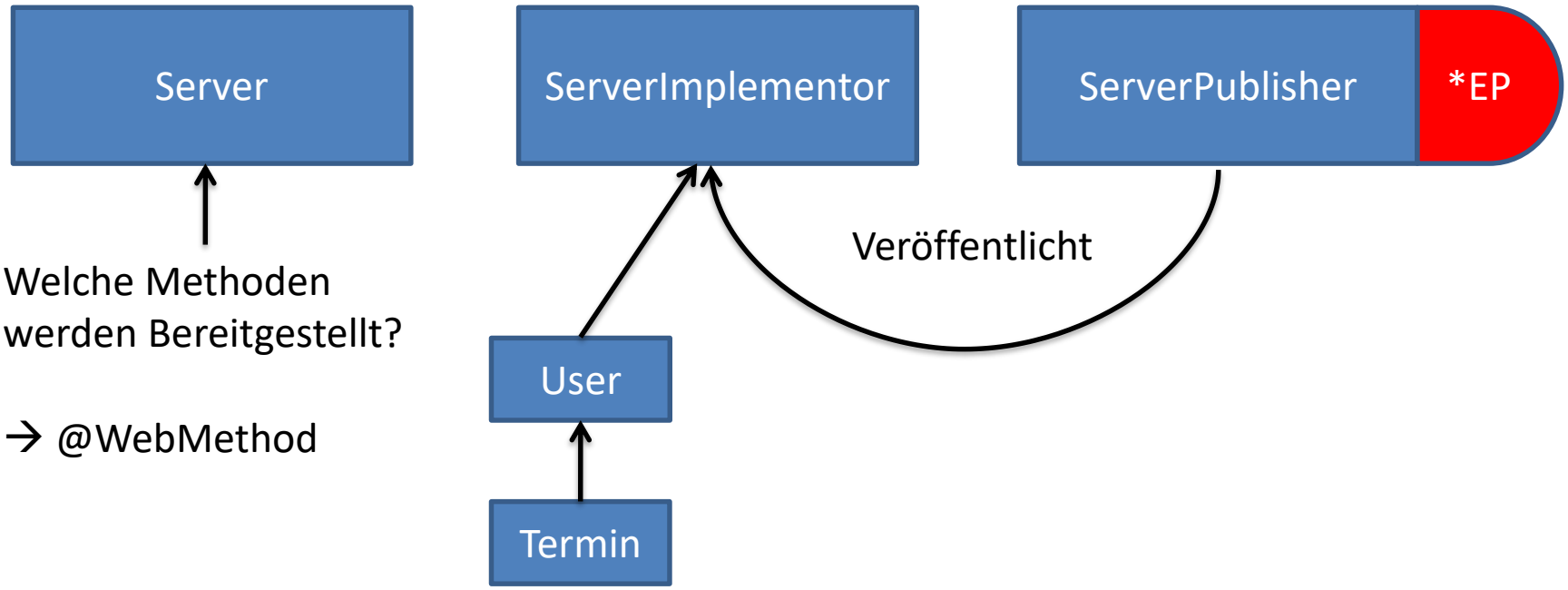
ServerImplementor.java

- implementiert das Interface Server
- enthält alle Funktionen die der WebClient aufrufen kann

ServerPublisher.java

- Veröffentlicht Endpunkt des Webservices
- Start, Stop, Restart Funktion

2.1 Struktur / Verbindung



2.2 Methoden zur Authentifizierung

- `public int login(String username, String passwort, String ip) { … }`
- Überprüft ob:
  - ein Benutzer mit dem übergebenen Namen existiert
  - das Passwort übereinstimmt
  - der Benutzer bereits aktiviert ist
- Wenn der Login erfolgreich war, wird dem User in der Datenbank die IP zugewiesen und ein neuer Zeitstempel gesetzt.

2.2 Methoden zur Authentifizierung

- `public boolean registration(String username, String passwort, String ip) { }`
- Überprüft ob:
  - ein Benutzer mit dem übergebenen Namen bereits existiert
- User wird in Datenbank aufgenommen und bekommt eine IP und ein neuer Zeitstempel zugewiesen

2.3 Methoden zur Administration

- `public boolean activateuser(String username, String passwort, String ip, String inactive_username) {}`
- Überprüft ob:
  - der Benutzer Administrationsrechte besitzt
  - die Registration existiert
- Falls beide Eigenschaften bestätigt sind, wird die gewünschte Registration aktiviert

2.3 Methoden zur Administration

- `public boolean deactivateuser(String username, String passwort, String ip, String active_username) {}`
- Überprüft ob:
  - der Benutzer Administrationsrechte besitzt
  - die Registration existiert
- Falls beide Eigenschaften bestätigt sind, wird die gewünschte Registration deaktiviert

2.3 Methoden zur Administration

- `public ArrayList<String> showNotActivatedUsers(String username, String passwort, String ip) {}`
- Überprüft ob:
  - der Benutzer Administrationsrechte besitzt
  - die Registration existiert
- Liefert alle Registrationen deren Status noch inactive steht

2.3 Methoden zur Administration

- `public boolean deleteuser(String username, String passwort, String ip, String registration) {}`
- Überprüft ob:
  - der Benutzer Administrationsrechte besitzt
  - die Registration existiert
- Falls beide Eigenschaften bestätigt sind, wird die gewünschte Registration aus der Datenbank gelöscht

2.4 Benutzerfunktionen

- ```
  public boolean
  changepasswort(String username,
  String altesPasswort, String ip,
  String neuesPasswort) {}
  ```
- Überprüft ob:
  - das aktuelle Passwort noch stimmt
- Datenbank übernimmt neues Passwort
  ```

2.4 Benutzerfunktionen

- `public boolean addTermin(String username, String passwort, String ip, String beginn, String ende, String thema, String ort, String wiederholung) {}`
- Überprüft ob:
  - die Loginangaben korrekt sind
- Termin wird in Kalender des Benutzers eingetragen

2.4 Benutzerfunktionen

- `public boolean removeTermin(String username, String passwort, String ip, int index) {}`
- Überprüft ob:
  - die Loginangaben korrekt sind
  - der Index des gewünschten Termins sich in dem erlaubten Bereich befindet
- Termin wird aus dem Kalender des Benutzers entfernt

2.4 Benutzerfunktionen

- `public ArrayList<String> showKalender(String username, String passwort, String ip) {}`
- Überprüft ob:
  - die Loginangaben korrekt sind
  - im Kalender ein Termineintrag existiert
- Liefert die komplette Terminliste

2.4 Benutzerfunktionen

- `public ArrayList<String> showKalenderFrom(String username, String passwort, String ip, String user) {}`
- Überprüft ob:
  - die Loginangaben korrekt sind
  - der andere Benutzer existiert
- Liefert die komplette Terminliste des gewünschten Benutzers

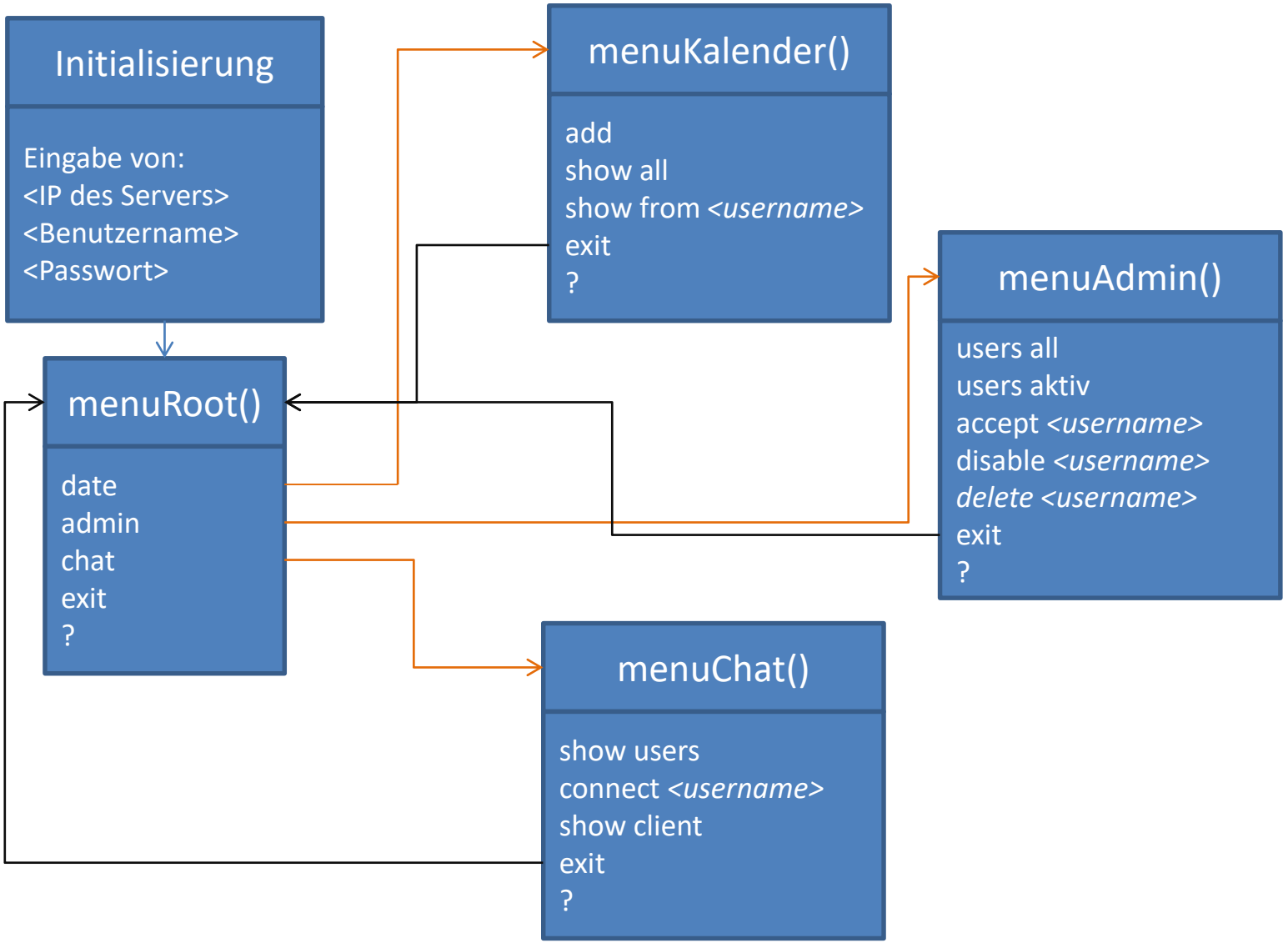
2.5 TCP Chat Server

- Auf jedem Client läuft ein TCP Chat Server im Hintergrund
- Dieser wartet an Port 1988 auf eine eingehende Verbindung
- Wenn eine Verbindung geöffnet wird öffnet der Chat Server das Chat Fenster

2.6 TCP Chat Client

- Der Client kann im Menü CHAT mit dem Befehl **connect <username>** die IP Adresse des Chatpartners erfahren und versuchen eine Verbindung zu dieser IP aufzubauen
- Wenn eine Verbindung geöffnet wird öffnet sich das Chat Fenster

3. Menü-Aufbau des Consolen Client



4. Menü-Aufbau des GUI Client

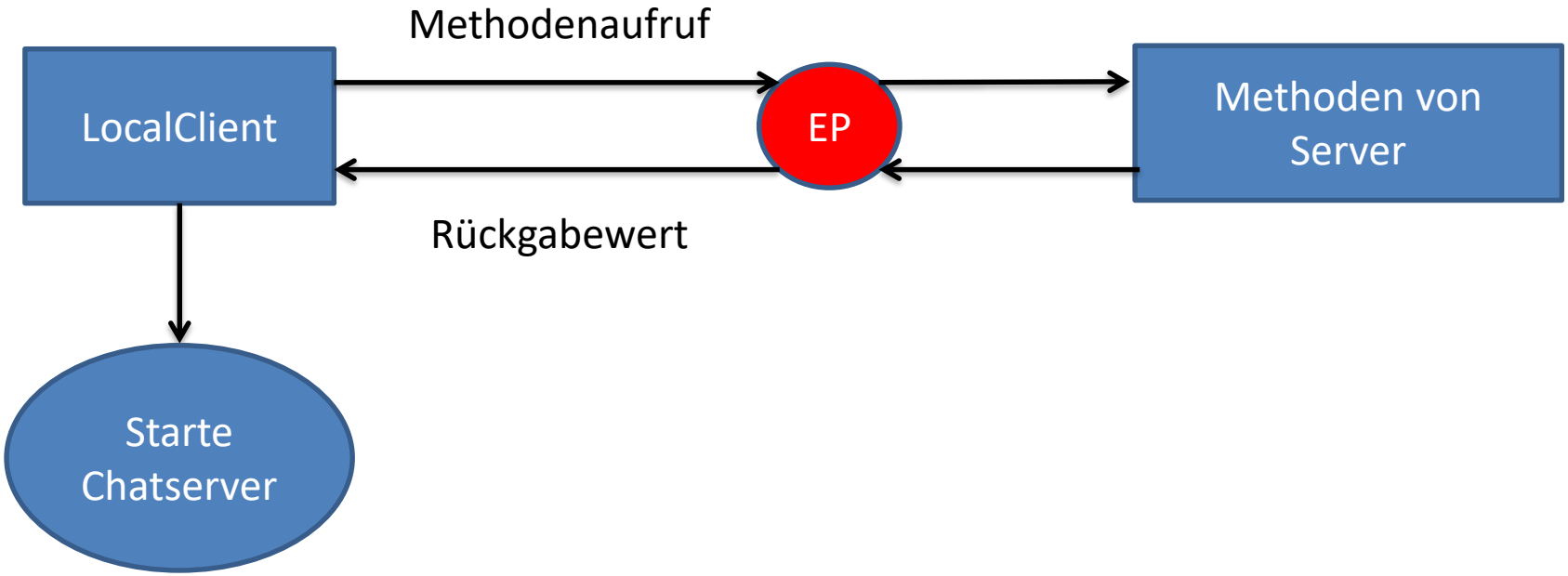
IP-Eingabe des Servers

Loginabfrage

Filemanager, Kalender, Homescreen usw.

Textbearbeitung

4. Struktur / Verbindung



- Methodenaufrufe werden mit JButtons und deren ActionListeners realisiert
- Chatserver wird im Aufbau des Clientframes gestartet.
- !Nur bestimmte Objekte können übergeben werden!

5. Dateistrukturen Client

Ordnerstruktur:

./username_clientfiles	Dateien der User
./username_clientfiles/Files	Standard Ordner
./Icons	Icons für den Filemanager

6. Dateistrukturen - Server

Ordnerstruktur:

./setupfiles/userlists	Userlisten
./setupfiles/calenderlists	Kalender
./serverfiles	Freigegebene Dateien

6. Dateistrukturen - Server

Userlisten:

./setupfiles/userlists

userlist.txt

Name		Passwort	Zeitstempel	Rechte	Aktiviert
aryan	:	layes	: 1328819298165 :	999 :	true;
arman	:	neysi	: 1328820030009 :	999 :	true;
bob	:	Baumeister	: 1328820078110 :	666 :	true;
mary	:	jane	: 1328820060419 :	666 :	true;

6. Dateistrukturen - Server

Kalender:

./setupfiles/calenderlists

calender_username.txt

Anfangszeit des Termins ;Ende des Termins ;Beschreibung ;Ort des Termins ;

Szenarien und Vorführung

Szenario 1

a. Mary loggt sich mit falschem Passwort ein

b. Mary erstellt eine neue Datei Info.txt…

c. Mary bearbeitet die Datei und lädt diese anschließend hoch (Dateifreigabe)…

d. Mary lädt sich eine Datei vom Server runter und bearbeitet diese…

Szenario 2

a. Mary bleibt eingeloggt

b. Bob loggt sich nun ein

c. Mary öffnet und bearbeitet eine Datei auf dem Server

d. Bob versucht auf die selbe Datei zuzugreifen

e. Bob informiert Mary über den Chat das sie die Datei schließen soll

f. Mary schließt Datei und meldet

Szenario 3

a. Mary editiert ihren Kalender

b. Bob will den Kalender von Mary einsehen

c. Bob versucht den Kalender zu bearbeiten