Diplomprojekt Ausarbeitung (inoffiziel)

Projektteam

Polak Rene
Projektleiter, Controlling

Morhammer Roland
Client-Server Architektur + Serverapp

Rus Alex
Webinterface & Datenbank + Serverapp

Erstellt von Schülern der 5/6 BBKIF als Diplomprojekt.
Themen: POS, MEDT, NVS
(Hauptthemen sind POS und MEDT)

1. Projektziel & Definition

Definition

Unser Projekt, der BackupMagican, ist eine Backupmanagmentsystem welche aus 2 großen Komponenten besteht:

Dem Server und Clients, dieses wird in einer Server-Client Architektur umgesetzt.
Ziel ist es Backups von einer Zentral, in diesem Fall der Server, zu steuern; Der Client ist der Zuhörer & Arbeiter
Um sich in das Netzwerk „einzubinden“ muss man sich entweder über einen Client oder über das Webinterface einen Account anlegen. Hat man die Clientsoftware noch nicht am Client installiert muss man dies nachholen.

Jeder Client der auf den angelegten User angemeldet ist und den Client mit dem User verknüpft hat, steht dem Server zu Verfügung, egal ob online oder offline. Der Server speichert sich sogenannte Aufträge und schickt sie wenn möglich an den Client.

Damit der Server immer weiß welche Client aktiv ist oder nicht authentifizieren sich die Clients beim Server, sie sagen einfach Hallo.

Sobald die Authentifikation abgeschlossen ist, der User eingeloggt ist, muss der User Clients zu seinem Account hinzufügen. Das passiert ganz einfach; Der Server merkt sich einfach die IP-Adresse des Clients, sobald dieser Vorgang abgeschlossen ist, kann der User gleich anfangen Aufträge zu schreiben. Man kann das entweder über das Webinterface machen oder lokal über den Client (greift auf das selbe Interface zu).

Aufträge sind kompakte JSON Files, zusätzlich gibt es auch normale Messages; Diese sind auch JSON Files. Diese Files werden über den body eines HTTP Packets zwischen Server und Host versendet.

Sobald der User seine Aufträge erstellt hat fängt der Server an mit den aktiven Clients zu kommunizieren und schickt diesen die Aufträge; Diese beinhalten Daten fürs Backup und Zugangsdaten zu Ziel-Host.
Der Client empfängt dieses und teilt das auch dem Server mit. Daraufhin verarbeitet der Client die Aufträge und teilt dies dem Server mit.

Der Server überprüft nun ob das Backup möglich ist (Ev. Cloud Speicher down, nicht genug Speicher etc), falls ja kriegt der Client das OK um die Daten an den Cloud-Service zu schicken.
Ist das Task fertig bearbeitet schickt der Client dem Server eine Nachricht das diese Task abgeschlossen ist. Falls nicht kriegt der Server einen LOG des Fehlers.

Das Webinterface selbst kann man auch über die Website ansprechen. Hierzu muss man sich nicht beim Client einloggen, wichtig ist nur das der Client online ist. Sobald dieser Umstand gegeben ist kann man Aufträge verteilen ohne Probleme.

Projektziele

1. Erstellung eines funktionalen Prototypen mit allen wichtigen Funktionen
2. Erstellung von Client-Backupsoftware
3. Erstellung von Webinterface
4. Erstellung von Serververwaltungssoftware
5. Kommunikations zwischen Server und Client
6. Senden von Files an Cloud Services

Nicht Ziele (Professionelle Ansätze):

1. Usermanagmentsystem für Authentifizierung (oder z.b OAuth)
2. Super ausgereifte Networksecurity
3. Extrem belastbare Server-software welche Millionen Nutzer aushält.

3. Technische Umsetzung

Die Struktur der einzelnen Komponenten

Client

Der Client besteht aus 2 Komponenten:

- Dem Backup Programm
- Dem Communicator

Backup Programm:

Das Backup Programm ist dafür zuständig aus JSON Aufträgen, welche es vom Communicator erhält zu verarbeiten.
Dazu erstellt es die Struktur in Objekten, liest die Files darin aus welche gespeichert werden soll und erstellt sich eine Liste.

Wenn dieser Vorgang abgeschlossen ist wird diese Liste dem Server mitgeteilt.
Der Server sagt dem Client nun ob das Backup möglich ist, wenn das der Fall ist wird das Backup auf den Ziel-Host gesendet.
Wenn nicht => Error Log am Server;

Zusätzlich kann der Client Default-Backups erstellen. Wie bei Aufträgen wird vorher ein Objekt-Baum erstellt; Dieser wird erst-gesichert und dann ein Timestamp gespeichert.
Wenn das Backup wieder ansteht wird überprüft ob sich etwas verändert hat;

Wenn ja => Backup erstellen und an Ziel-Komponente schicken
Wenn Nein => Warten bis das nächste Intervall abgelaufen ist.

Communicator:

Der Communicator spricht mit dem Server über TCP; Er sendet oder empfängt HTTP Packages.
Als erstes Authentifiziert er sich beim Server und sagt ihm damit „Hallo“;

Hat der Server einer Aufgabe teilt er diese dem Client mit; Ansonsten arbeitet der Client seine Default Backups ab, sofern welche enthalten sind.

Der Client kommuniziert auch mit den Ziel-Hosts und schickt die gewünschten Daten an diesen. Daher benötigt er auch Zugangsdaten für die Übertragung welche er mit den Aufträgen vom Server bekommt. Sobald der Auftrag verarbeitet ist wird überprüft ob das Backup möglich ist, wenn ja baut der Communicator eine Verbindung zum Ziel auf (Oauth bei Cloudspeichern).

Er schickt dann mit den Zugangsdaten die Daten auf den Cloudspeicher / Ziel-Host)

Server

Der Server besteht aus 2 Komponenten

- Communicator
- Hauptprogramm

Communicator

Der Communicator funktioniert ähnlich wie der gleichnamige Communicator des Clients. Unterschied ist das der Server Aufträge verschickt und Rückmeldungen bzw. Authentifizierungen empfängt und bestätigt.

Der Communicator ist also Bindeglied zwischen Client & Server.
Kommunizieren wird der Communicator des Servers, genau so wie der Client Communicator, auch über HTTP Pakete welche JSON Files im Body enthalten werden.
Der Communicator wird verschiedene Arten von JSON Anfragen verarbeiten können

Hauptprogramm

Das Hauptprogramm ist zuständig dafür das die HTTP Pakete die am Server ankommen verarbeitet werden. Es erstellt Communicator; Für jeden Client-Communicator gibt es dann einen Server-Communicator. Zusätzlich ist das Hauptprogramm für die Authentifizierung zuständig und es füllt auch Userdaten in die JSON Files ein.

Das Hauptprogramm und die Communicatoren sind in einem großen Programm enthalten.
Zusätzlich hat das Hauptprogramm die Aufgabe, Fehler LOGS zu erstellen, sicherstellen das die Communicator funktionieren usw.

Webinterface & Datenbank & Website

Webinterface:

Das Webinterface dient dazu dem User die Möglichkeit zu geben Aufträge zu erstellen, Default Backups zu bearbeiten und Userdaten zu ändern.
Um auf dieses zugreifen zu können muss der User sich registrieren und anmelden.
Sobald dieser Vorgang abgeschlossen ist kann er auf das Webinterface zugreifen.
Dieses besteht aus einer Raster-Ansicht.

Zuerst wählt der User aus welchen Client er nutzen will.
Dann wird die gesamte Baumstruktur des Clients angezeigt (Baumstruktur sind alle Ordner).
Der user kann diese dann auswählen und wenn er alle Ordner definiert hat kann er diese Einstellung übernehmen. Der User teilt dem Webinterface dann nur mehr mit welche Dateiendungen er speichern will.

Dannach schickt das Webinterface das JSON File (Auftrag) an den Server und dieser schickt dieses mit den Authentifizierungsdaten an den Client.

Datenbank:

Speichert alle Daten der User.

- Userdaten des BackupMagicans
- CloudUser Daten
- Pfad zu den Strukturen der Clients
- Clients & User Beziehungen

Website:

Ist die Präsentationsseite.

Bringt das Programm dem User nahe und zeigt ihm die Nutzen und Vorteile unserer Software.
Wird Repsonsive sein. User kann gleich über Website anmelden und den Client runterladen.

Technologien:

- **QT (C++) für Server Programm und Client Programm**
  Entwicklungsumgebung: QT Creator

- **HTML5, CSS3, JavaScript**
  Entwicklungsumgebung: Notepad++

- **Mysql Datenbank**
  Speichert relevante Userdaten

- **TCP/IP bzw. HTTP**
  Wird in Communicatoren benutz

- **JSON Files**
  Kommunikationsmittel zwischen Komponenten; Cloud Services auch

- **Cloud-Services (Google Drive, Icloud, Dropbox)**
  Zugangsdaten sind in MYSQL Datenbank

- **Git (2. Semster für Versionkontrolle)**
  GitHub-Client wird für Verwaltung des Projekts genutzt

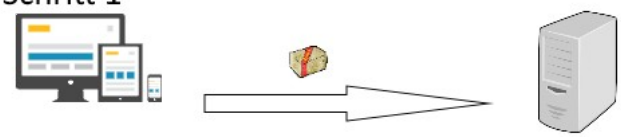
- **(Eventuell Maven für Projekt-Building)**
  Wird aktiv in die Entwicklungsumgebung eingebunden

4. Grapfische Darstellung der Abläufe

Client meldet sich beim Server

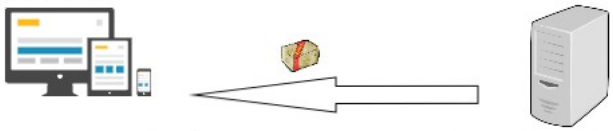
BackupMagican Loggt sich am Server ein

Schritt 1



Client sendet HTTP Packet mit JSON File im Body; Dieses enthält User + Passwort und Client ID

Schritt 2



Server schickt entweder OK oder inkorekte Userdaten zurück.
Falls die Daten nicht stimmen muss Schritt 1 von Client nochmal wiederholt werden; Ansonsten trägt der Server den Client als aktiv und nutzbar ein.

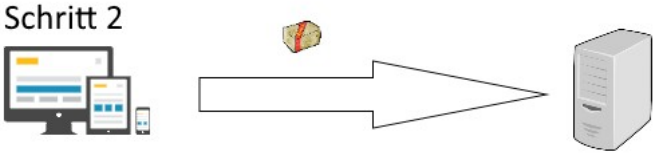
Lebenszyklus eines Auftrags

Erstellung und Verarbeitung eines Auftragess

Schritt 1

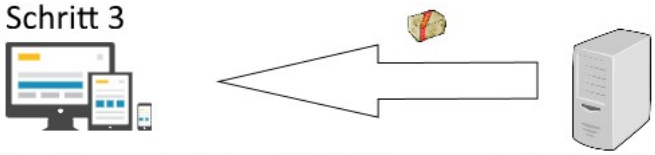
Auftrag wird erstellt; Entweder in der Handyapp, am Client oder über das Web interface;

Schritt 2



Unter der Annahme das der Client eingeloggt ist schickt er über HTTP den Auftrag. Dieser ist im Dateiformat JSON; Der Server nimmt dieses hingegen und fängt dann an dieses zu verarbeiten

Schritt 3

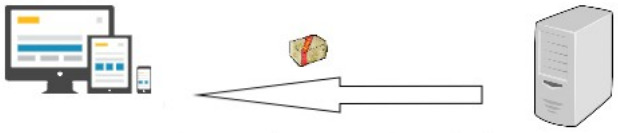


Der Server hat das JSON File verarbeitet und überprüft und leitet es jetzt an den Client los auf dem die Sicherung durchgeführt werden muss. Falls dieser nicht aktiv ist wird der Auftrag in Evidenz gehalten bis sich der Client meldet und wieder online bzw. aktiv wird.
Hat der Client das Paket erhalten startet die Auftragsverarbeitung

Client verarbeitet einen Auftrag vom Server-Client

Client verarbeitet Auftrag von Server

Schritt 1

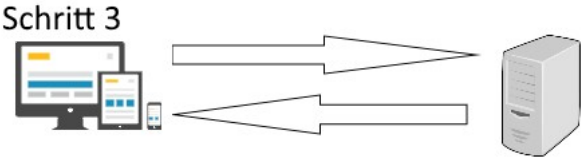


Server sendet Auftrag an den definierten Client. Der Client ist in seiner aktiven Zeit auf Wartemodus; Das heißt das er auf einen Auftrag vom Server wartet. Der user kann auch Defaul-Sicherungen erstellen, diese werden fertig gestellt (falls vorhanden). Wenn keine vorhanden sind nimmt er das Packet an und verarbeitet es.

Schritt 2

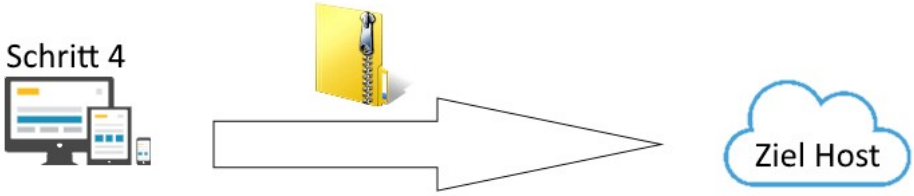
Client verarbeitet JSON File mit seinem internen Backup-Programm

Schritt 3



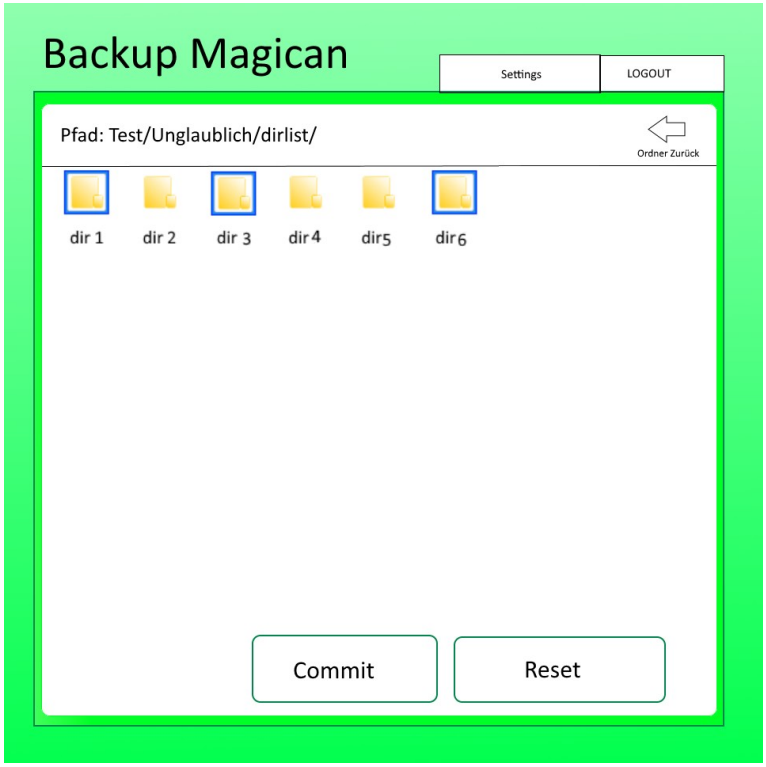
Client und Server kommunizieren um fest zustellen ob Backup möglich ist. Falls nicht wird es verworfen

Schritt 4



Client sendet nun Daten an den Zielhost; Die Informationen hat er davor noch vom Server erhalten wie er sich über die Möglichkeit der Sicherung erkundigt hat Wenn Client fertig ist teilt er dies dem Server nur noch schnell mit über ein HTTP JSON File welches dem Server über den Umstand informiert

Webinterface Beispiel 1



Die Website bietet Standardfunktionen wie Optionen und Logout; Das Layout ist wie bei einem Explorer.

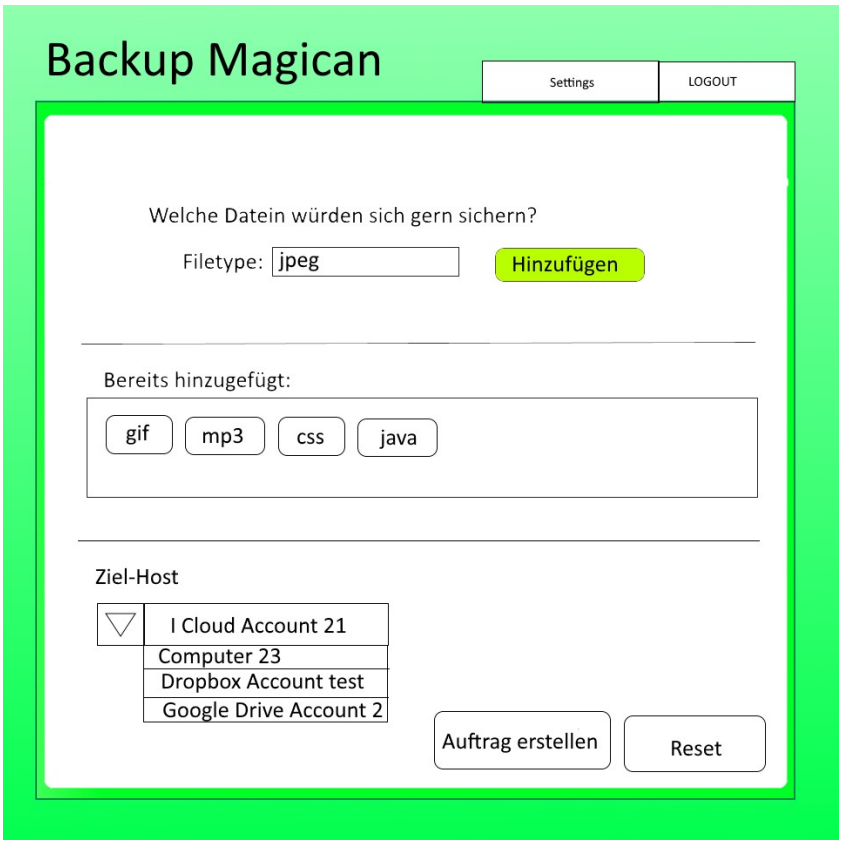
Es wird der aktuelle „Pfad“ angezeigt und die Option angeboten einen Ordner hinauf zu springen.

Ordner sind im Raster angeordnet.

Ein klick markiert diesen mit einen blauen Rand, das heißt das diese für die Sicherung aufgenommen werden, klickt der User nochmal demarkiert er diesen Ordner. Bei 2 Klicks springt er in den Unterordner;

Commit heißt das die Ordner auswahl abgeschlossen ist; Reset setzt das Interface zurück

Webinterface Beispiel 2



Es gibt 4 Abschnitte:

1. User kann über die File Maske neue Dateiarten für die Sicherung hinzufügen;

2. Die Liste der bereits hinzugefügten Dateiendungen; nach diesen wird bei der Sicherung in den Ordner gesucht.

3. Dropdown Menu für Zielhosts. Dieses ermöglicht es den User auszusuchen auf welchem Host er die Sicherung ausführen lassen will.

4. Auftragerstellen: Damit gibt der user bekannt das sein Auftrag fertig ist und ausgeführt werden soll; Reset setzt die Maske der jetztigen Page des Interfaces zurück.

5. Derzeitiger Stand

Server

Roland (im 2ten Semster + Polak & Rus)
Es ist zurzeit ein Webserver vorhanden, welcher die JSON Files aus den Anfragen des Clients heraus lesen kann; Zurzeit ist das eine externe Library die wir erweitert haben.

Im nächsten Semester schreiben wir das Hauptprogramm am Server welches die Vorgänge steuert und lenkt, dann ersetzt wir das Kommunikationstool (Communicator) mit einem eigens geschriebenen Programm.

Client

Rene:
Das Backup Programm steht; Es muss noch der JSON Parser fertig geschrieben werden und die SIGNALS und SLOTS fertig gestellt werden. Diese sind abhängig von Client-Communicator.

Roland:
Client-Communicator kann schon JSON Files über HTTP Packete an den Server senden und empfangen. Client schickt user + passwort, Server überprüft das und schickt als Antwort entweder erfolgreich oder fehlschlag.

Webinterface & Datenbank

Rus:
Webinterface ist gerade in Arbeit. Es soll ein Raster sein, welches Ordner & Files anzeigt.
User kann mit links klick Daten & Ordner markieren, Doppelklick geht einen Ordner tiefer.
Es gibt auch noch einen „return“ Knopf der einen Ordner rauf wandert und ein commit Knopf welcher den Auftrag erstellt.

Datenbank für Userdaten steht schon (Rudimentär; Es fehlen noch Projektspezifische Daten wie verfügbare Clients und Ordner Strukturen)

Website für Programm

Polak:
Design technisch ist die Website schon fertig; Sie beinhaltet selbst erstellte Grafiken, Erklärungen zum Projekt und Ziele und wirbt dieses auch an.
D.h. Website ist gleich auf „Verkaufsort“ der Idee. User können dann im finale Produkt die Website durchsuchen, und sich dann gleich von dieser über das Webinterface anmelden und die Software nutzen.

Es fehlt einfach noch mehr Inhalt und Einbindung des Webinterfaces.