



Jarvis als Schnittstelle zwischen dem ATM und einem Referenzrechner

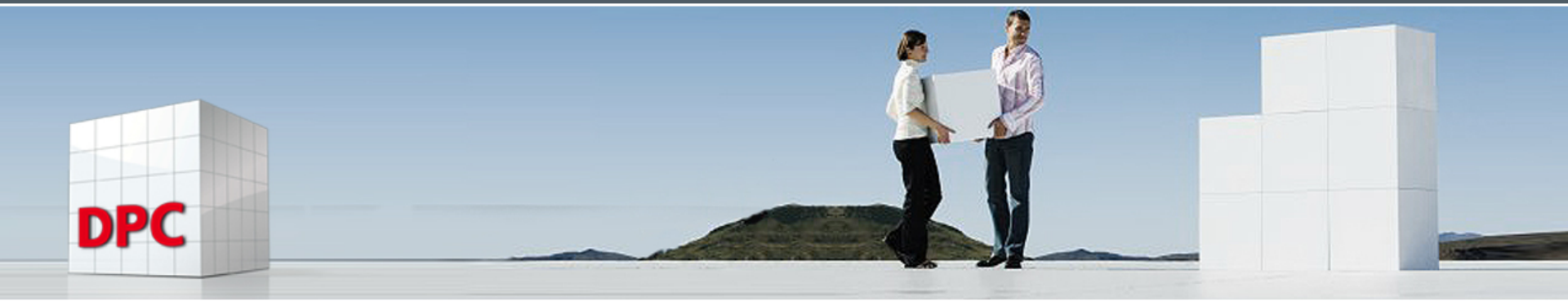
Alexander Krämer
Solingen, 07.11.2024



Jarvis als Schnittstelle zwischen dem ATM und einem Referenzrechner

Inhaltsübersicht

1. Was ist Jarvis?
2. Warum Jarvis?
3. Was ist zu tun?
 1. Aufgaben & Herausforderungen
 2. Grundlegende Architektur – Einzel-, Massen- & Großtest
 3. Konfiguration
4. Demo

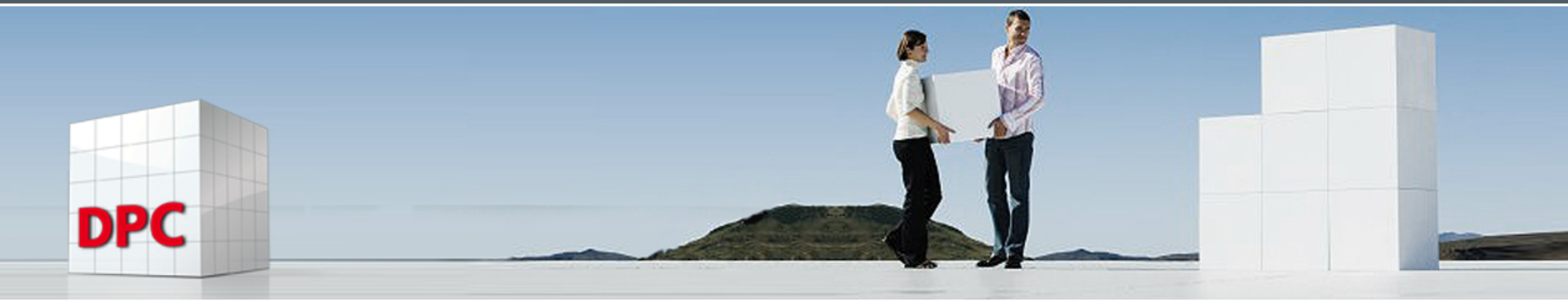


Was ist Jarvis?



Jarvis – JSON and REST Service

- Conga-basierter Webserver; erwartet HTTP(S)-Post-Requests (und sendet Responses) mit Payload im JSON-Format
- entwickelt (hauptsächlich) von Brian Becker seit 2017 (ursprünglich als „JSONServer“)
- auf GitHub unter der MIT-Lizenz verfügbar
- im Code: Klasse

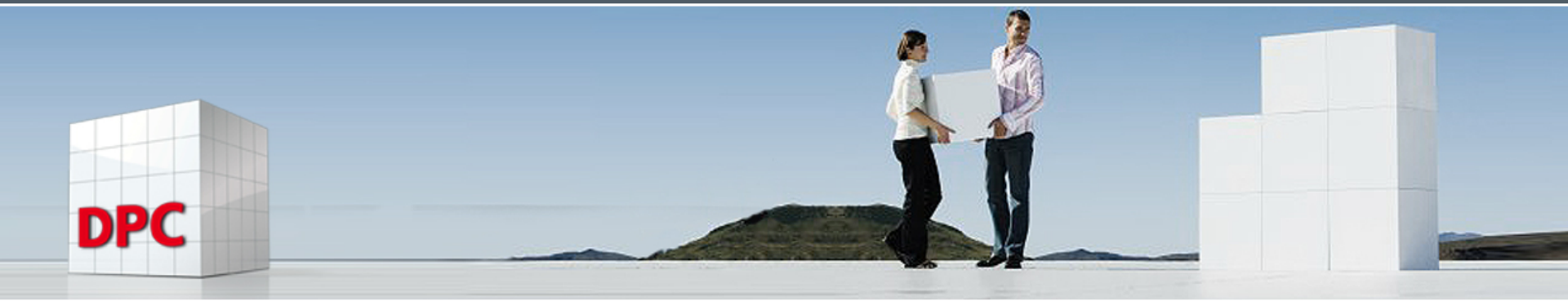


Warum Jarvis?

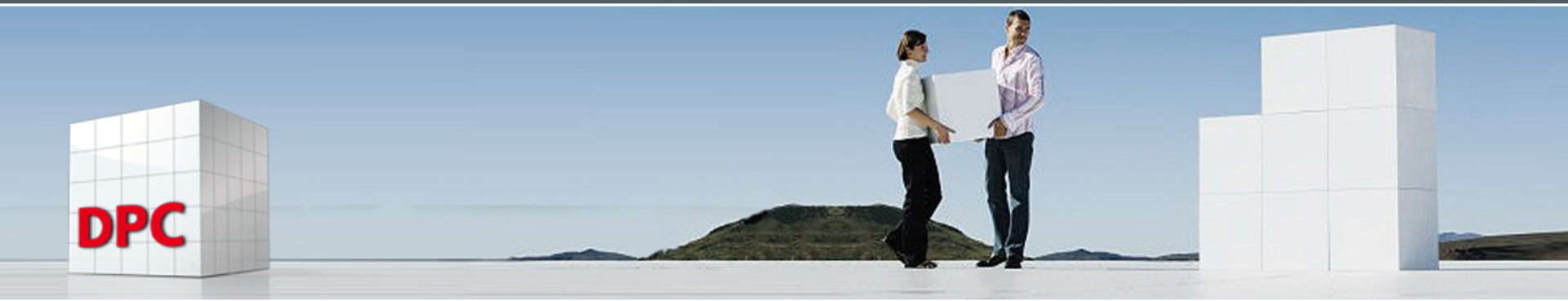


Vorteile von Jarvis als Schnittstelle

- einfach einzustellen und zu nutzen – Bereitstellung im lokalen Netzwerk ohne Weiteres möglich
- verwendet Standardformat JavaScript Object Notation für In- und Output
- ohne zusätzliche Lizenzgebühren
- vielfältige Konfigurationsmöglichkeiten
- strikte Trennung zwischen ATM und APL (oder allgemein Client und Server) möglich
- mit zusätzlichen Tools (Docker, Cloud) erweiterbar



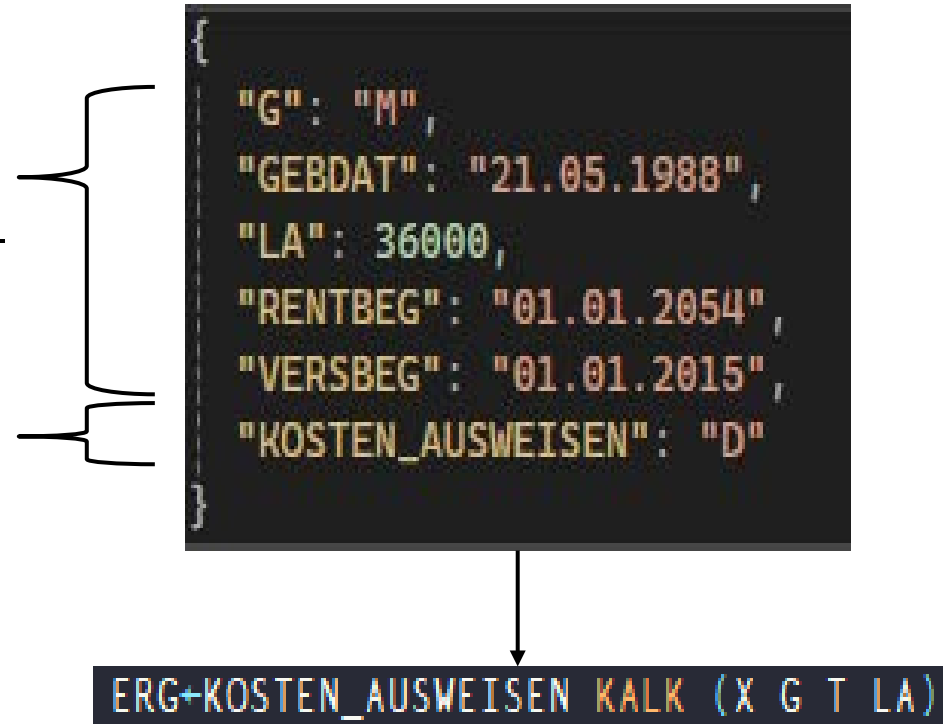
Was ist zu tun?



Aufgaben & Herausforderungen

Was geht rein?

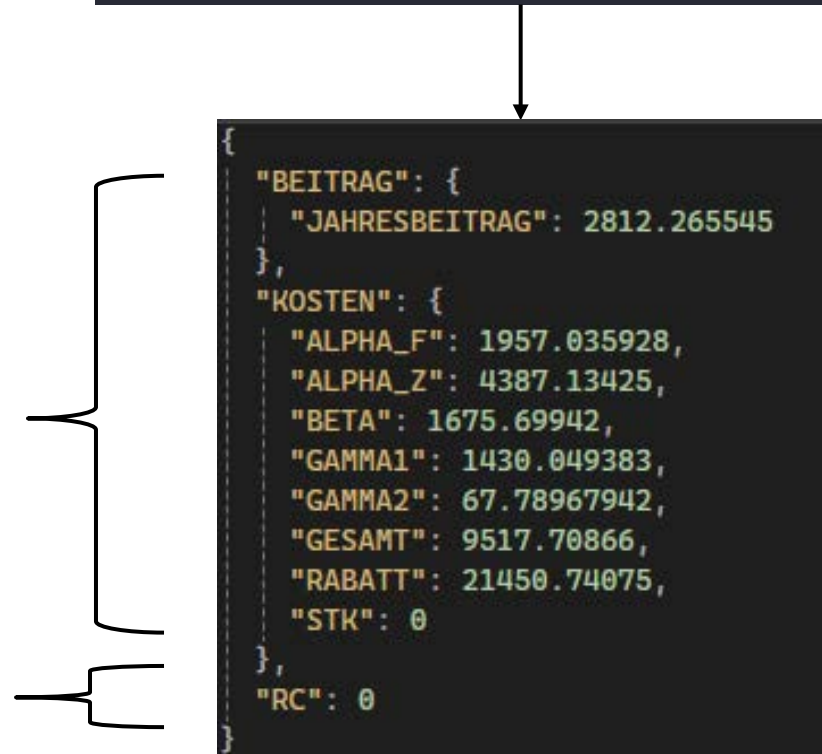
- Berechnungsparameter
verbundene Frage: Wie passt die Datenstruktur für den RR in JSON?
zu erwarten: ER-Inputstruktur, z. B. LF-Datenmodell
- technische Parameter (z. B. Angaben dazu, was zu berechnen ist)
hängt stark vom Einzelfall ab
- eventuell weitere, neue Parameter



Was kommt raus?

- Berechnungsergebnisse
alternativ ggf. Fehlerinformationen
auch hier Frage nach passender
Struktur für die JSON-Ausgabe
zu erwarten: ER-Outputstruktur, z. B.
LF-Datenmodell
- Returncode

ERG+KOSTEN_AUSWEISEN KALK (X G T LA)

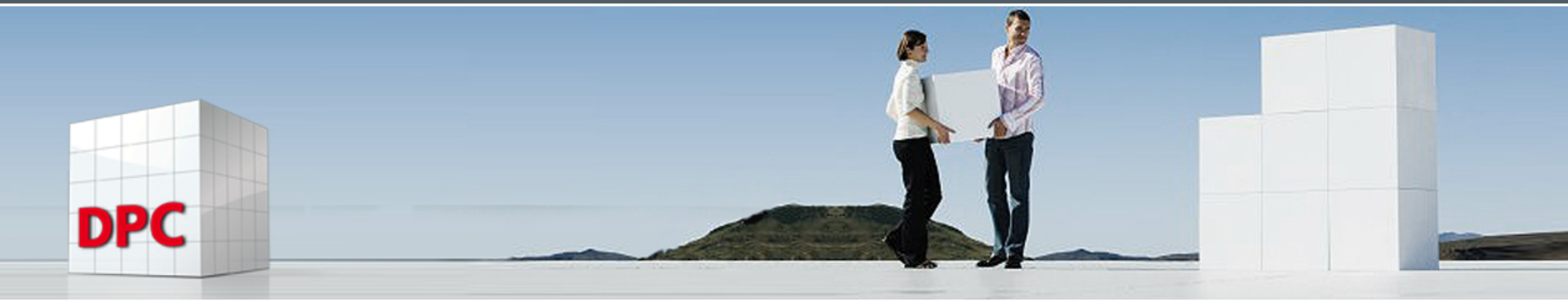


Sonstige Jarvis-Aufgaben

- Konfiguration, dabei teilweise Abstimmung mit der IT nötig
 - Auswahl/Reservierung von Ports
 - Sicherheitsentscheidungen: Authentifizierung, gesicherte Verbindung, „Firewall“
- Sind weitere Schritte nötig: Start- & Stopfunktion, Hintergrundtasks?
- ggf. automatisierter Start multipler Server

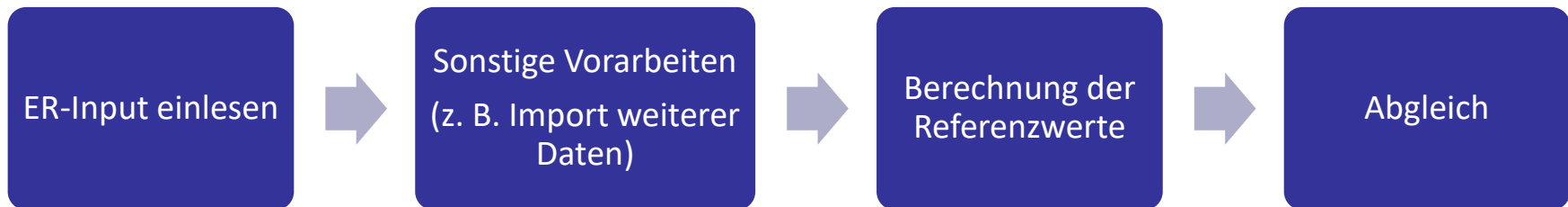
Einbettung in den Code

- Anspruch an Inputparameter der Berechnung: nicht nur inhaltlich passend, sondern auch korrekt formatiert – nicht jede APL-Datenstruktur ist „JSON-fähig“
- alte Schritte auslassen oder neue hinzunehmen – Verdrahtung zwischen dem, was weg soll, und dem, was bleibt?
- Wie erkennt man ggf. zur Laufzeit, ob der Aufruf von Jarvis kommt?
- Multithreading-Probleme (z. B. bei globalen Variablen)
- Debugging und Errortrapping (über den Standard hinaus)

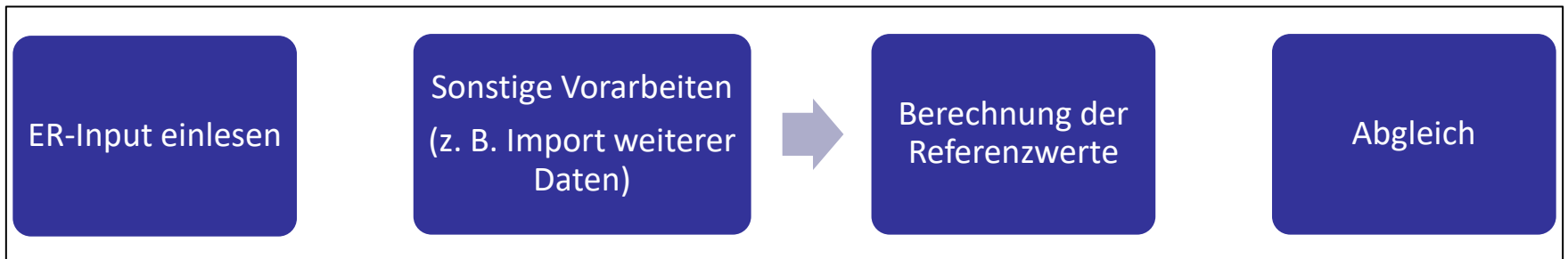
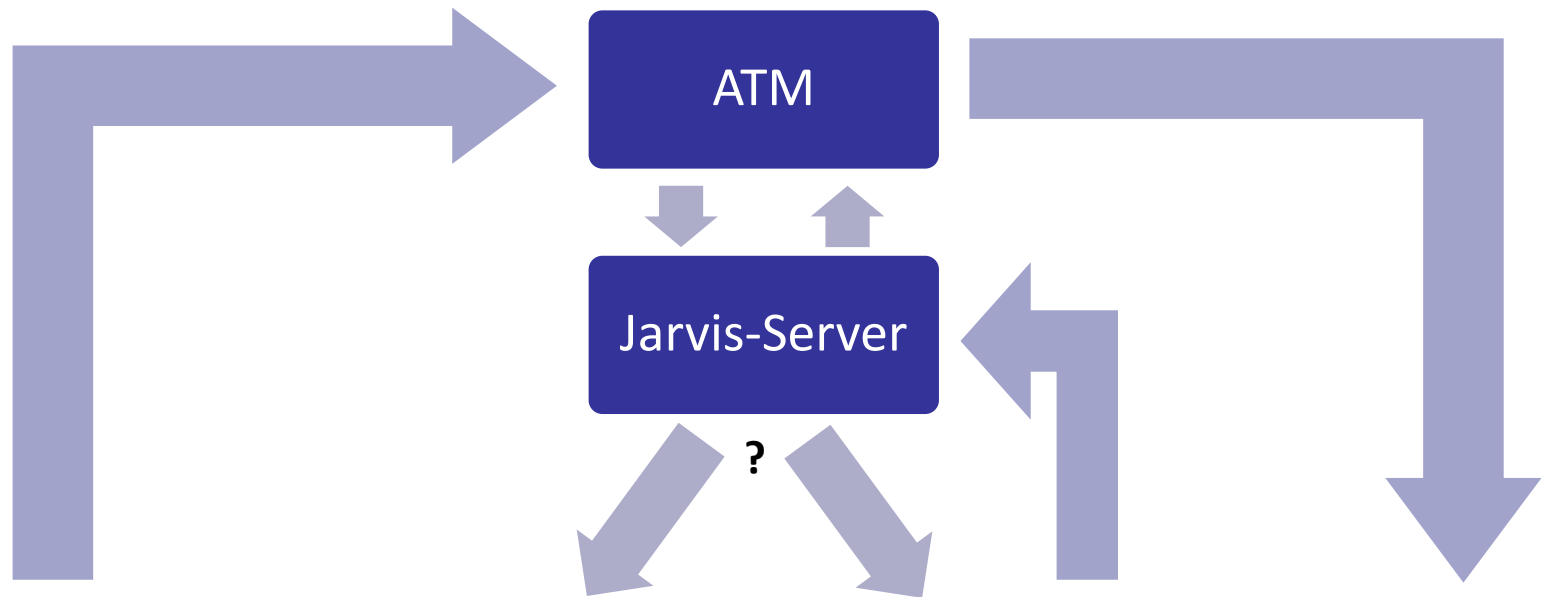


Grundlegende Architektur Einzel-, Massen- & Großtest

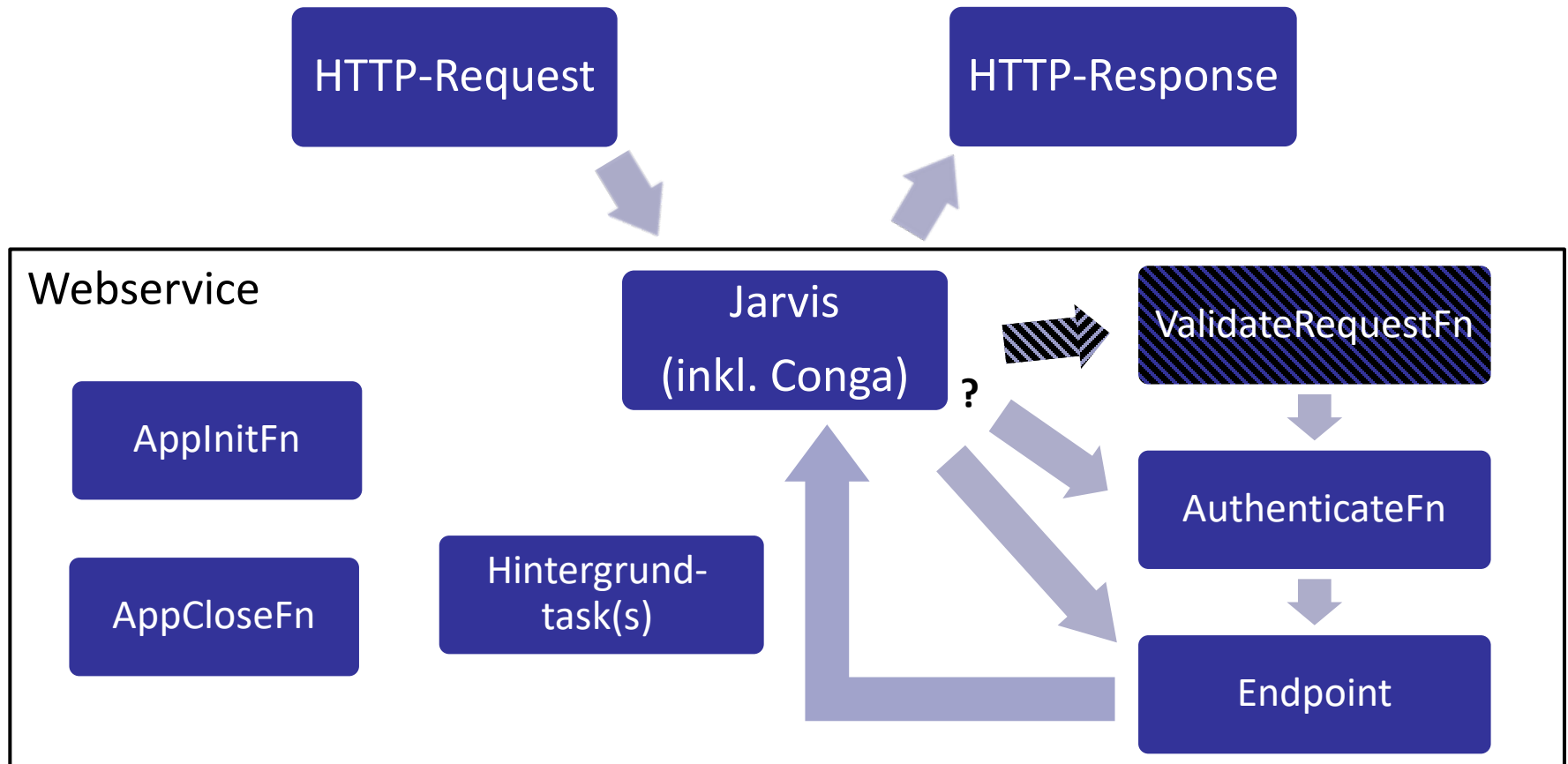
Typischer Ablauf - komplett im Referenzrechner



Mit dem ATM – klare Architektur und Schnittstellen



Ablauf im Webservice



Endpoints – Einzel- und Massentest

■ JARVIS_EINZEL (Einzelberechnung)

- Errortrapping und –handling (Zeilen 10 bis 27)
- Jarvis-bezogene Anpassungen (z. B. Lokali-sierung globaler Variablen etc.; Zeile 13)
- applikationsbezogene Anpassungen (z. B. Ableitung von Inputparametern; Zeile 15)
- Aufruf der Berechnungshauptfunktion (Zeile 18)
- Jarvis-bezogene Formatierung der Ausgabe (Zeilen 30 bis 34)

■ JARVIS_SEQUENZ (Massenberechnung)

- „JARVIS_EINZEL + Each“
- für Performanceoptimierung

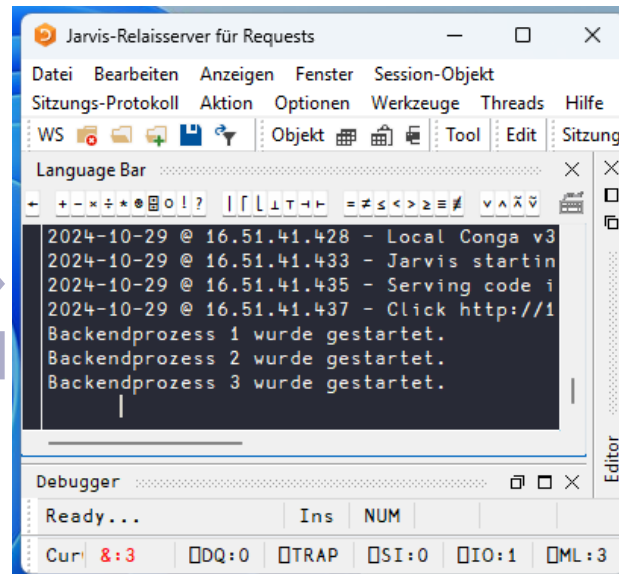
```
[0] ERG+JARVIS_EINZEL INPUT;InputIsString;APPL;APPL_ERG;RC;Input_ErgebnisFormatieren
[1] Ao DPC/AK,11.04.2024; DPC/AK,30.04.2024
[2] Ap Jarvis-Endpoint für die Einzelbearbeitung von Requests
[3] Ad INPUT Payload des Requests an den Webserver;
[4] Ad enthält die versicherungsmathematischen Parameter für die Beit
[5] Ad InputIsString boolescher Indikator, ob der Input als String oder als NS anko
[6] Ad ERG Rückgabedaten des Webservers, im gleichen Format wie die Eing
[7] Ag ###.Appl, ΔDEBUG
[8] As ###.Appl.KALK, Utilities_Jarvis.(ΔJSON ErgebnisFormatieren ApplAnpassungen.Ab
[9]
[10] :Trap (##.ΔDEBUG≠1)/0 ATrap wird im Debug-Modus (ΔDEBUG=1) deaktiviert
[11] INPUT+##.Utilities_Jarvis.ΔJSON*(InputIsString+326=□DR INPUT))INPUT
[12]
[13] APPL+□NS ###.Appl AErstellen einer Kopie des Namespaces, in dem die Berechnu
[14]
[15] INPUT+##.Utilities_Jarvis.ApplAnpassungen.AbgeleiteteParameter INPUT
[16]
[17] :Trap (##.ΔDEBUG≠1)/0 ATrap wird im Debug-Modus (ΔDEBUG=1) deaktiviert
[18] APPL_ERG+INPUT.KOSTEN_AUSWEISEN APPL.KALK INPUT.(X G T LA)
[19] RC+0
[20] :Else
[21] RC+1
[22] APPL_ERG+□DMX.(DM EN ENX Message)
[23] :EndTrap
[24] :Else
[25] RC+2
[26] APPL_ERG+□DMX.(DM EN ENX Message)
[27] :EndTrap
[28]
[29] AErgebnis formatieren und dann zurückgeben
[30] 'Input_ErgebnisFormatieren' □NS 'APPL_ERG' 'RC' 'InputIsString'
[31] :If 2=INPUT.□NC 'TFKEY' ABesitz der Datensatz ein Schlüsselattribut? Dann wird es
[32] 'Input_ErgebnisFormatieren' □NS 'INPUT.TFKEY'
[33] :EndIf
[34] ERG+##.Utilities_Jarvis.ErgebnisFormatieren Input_ErgebnisFormatieren
```

Load Balancing mit Front- und Backends

Frontend

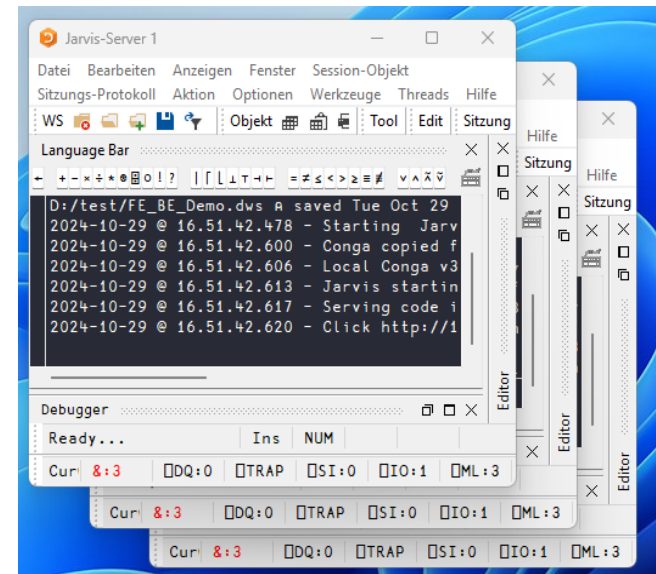
Backends

ATM



```
2024-10-29 @ 16.51.41.428 - Local Conga v3
2024-10-29 @ 16.51.41.433 - Jarvis startin
2024-10-29 @ 16.51.41.435 - Serving code i
2024-10-29 @ 16.51.41.437 - Click http://1
Backendprozess 1 wurde gestartet.
Backendprozess 2 wurde gestartet.
Backendprozess 3 wurde gestartet.
```

Endpoint: JARVIS_FRONTEND
(reine Relaisfunktion)

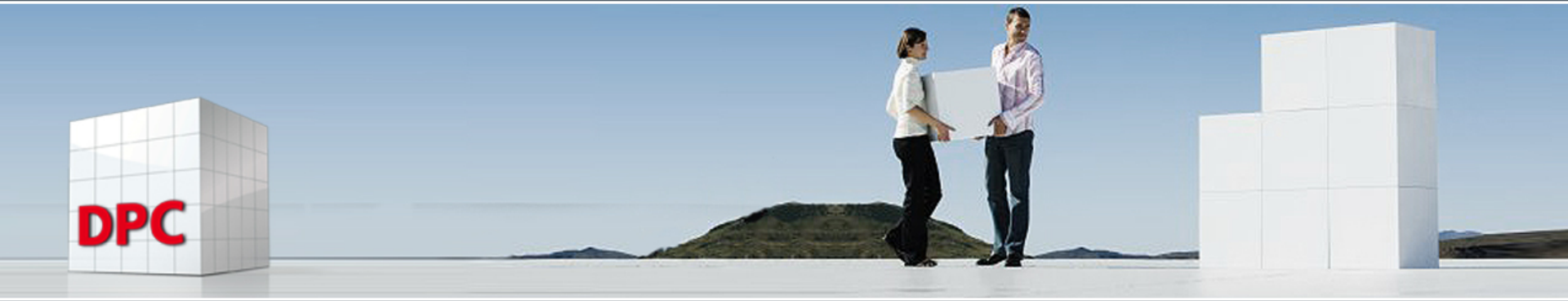


```
D:/test/FE_BE_Demo.dvs A saved Tue Oct 29
2024-10-29 @ 16.51.42.478 - Starting Jarv
2024-10-29 @ 16.51.42.600 - Conga copied f
2024-10-29 @ 16.51.42.606 - Local Conga v3
2024-10-29 @ 16.51.42.613 - Jarvis startin
2024-10-29 @ 16.51.42.617 - Serving code i
2024-10-29 @ 16.51.42.620 - Click http://1
```

Endpoint: JARVIS_EINZEL,
JARVIS_SEQUENZ
(Berechnungsfunktion)

Was läuft „nebenher“?

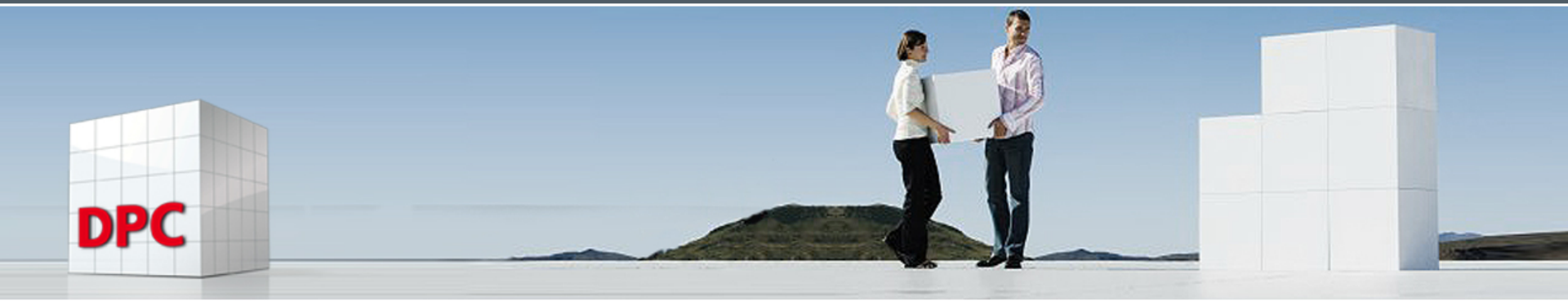
- laufende Wartungsaufgaben rund um Jarvis, die keinen menschlichen Input benötigen (sollen)
- läuft „im Hintergrund“ (d. h. Endlosschleife im eigenen Thread) und muss beim „echten“ Stopp des Webservice auch beendet werden
- Anwendung z. B. automatischer Neustart nach Absturz
- je nach Anwendung/Kundenwunsch weitere Aufgaben, z. B. täglicher Git-Pull



Konfiguration

Konfigurationsoptionen in Jarvis

- sicherheitsbezogen
 - Authentifizierung: AuthenticateFn
 - HTTPS-Protokoll: Secure, ServerCertFile, ServerKeyFile
 - IP-Adressen blocken: AcceptFrom, DenyFrom
- Größe der Massentestpakete: BufferSize, DOSLimit, WaitTimeout
- weitere Möglichkeiten, z. B.
 - Einrichtung von Sessions: SessionInitFn, SessionTimeout, ...
 - Ermöglichen von Get-Requests: AllowGETs



Vielen Dank für Ihre Aufmerksamkeit bisher!

Jetzt folgt noch die Live-Demonstration.